



COMPARAÇÃO E CLASSIFICAÇÃO DE SEQUÊNCIAS DE PROTEÍNAS
UTILIZANDO MINERAÇÃO DE TEXTOS

Carla Corrêa Tavares dos Reis

Tese de Doutorado apresentada ao Programa de Pós-graduação em Engenharia de Sistemas e Computação, COPPE, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Doutor em Engenharia de Sistemas e Computação.

Orientador(es): Susana Sheimberg de Makler

Nelson Francisco Favilla Ebecken

Rio de Janeiro

Julho de 2011.

COMPARAÇÃO E CLASSIFICAÇÃO DE SEQUÊNCIAS DE PROTEÍNAS
UTILIZANDO MINERAÇÃO DE TEXTOS


Carla Corrêa Tavares dos Reis

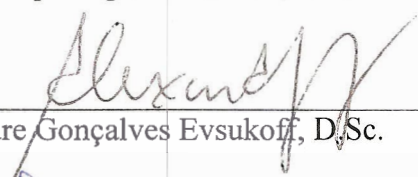
TESE SUBMETIDA AO CORPO DOCENTE DO INSTITUTO ALBERTO LUIZ
COIMBRA DE PÓS-GRADUAÇÃO E PESQUISA DE ENGENHARIA (COPPE) DA
UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS
NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE DOUTOR EM CIÊNCIAS EM
ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

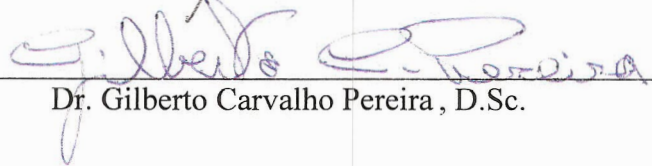
Examinada por:


Prof. Susana Sheimberg de Makler, D.Sc.


Prof. Nelson Francisco Favilla Ebecken., D.Sc.


Dr. Jonas Enrique Aguilar Perales, D.Sc.


Prof. Alexandre Gonçalves Evsukoff, D.Sc.


Dr. Gilberto Carvalho Pereira, D.Sc.

RIO DE JANEIRO, RJ - BRASIL

JULHO DE 2011

Reis, Carla Corrêa Tavares dos

Comparação e Classificação de Sequências de Proteínas utilizando Mineração de Textos/ Carla Corrêa Tavares dos Reis – Rio de Janeiro: UFRJ/COPPE, 2011.

IX, 121 p.: nil.; 29,7 cm.

Orientadores: Susana Sheimberg de Makler.

Nelson Francisco Favilla Ebecken

Tese (doutorado) – UFRJ/ COPPE/ Programa de Engenharia de Sistemas e Computação, 2011.

Referências Bibliográficas: p. 115-121.

1. Mineração de Textos. 2. Alinhamento de Sequências Moleculares. 3. Categorização Bayesiana. 4. Categorização Linear I. Makler, Susana Sheimberg de. et al. II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia de Sistemas e Computação. III. Título.

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

COMPARAÇÃO E CLASSIFICAÇÃO DE SEQUÊNCIAS DE PROTEÍNAS
UTILIZANDO MINERAÇÃO DE TEXTOS

Carla Corrêa Tavares dos Reis

Julho/2011

Orientadores: Susana Sheimberg de Makler

Nelson Francisco Favilla Ebecken

Programa: Engenharia de Sistemas e Computação

O objetivo deste trabalho busca avaliar as ferramentas de mineração de textos para categorização de cadeias de caracteres utilizando exemplos reais, no caso sequências de aminoácidos. Para atender este propósito foi desenvolvida uma metodologia capaz de treinar um classificador de textos sobre algumas das sequências de proteínas, e utilizarmos este classificador treinado para prever a categoria para as demais sequências não identificadas. Ou seja, identificar padrões de ordenação dos aminoácidos para algumas das sequências, os quais, considerando sua ocorrência por repetidas vezes, consigam representar uma categoria ou classe. Em seguida, o classificador é projetado sobre as sequências originais para determinar as regiões similares das sequências. A partir deste ponto, demonstra-se eficácia na abordagem de predição da localização de regiões específicas dessas proteínas que justifiquem os alinhamentos possíveis entre as sequências, uma vez que os classificadores identificam e categorizam com sucesso as sequências que compartilham regiões similares. Os resultados experimentais observados corroboram os resultados previstos analiticamente.

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

COMPARISON AND CLASSIFICATION OF PROTEIN SEQUENCES USING TEXT MINING

Carla Corrêa Tavares dos Reis

July/2011

Advisors: Susana Sheimberg de Makler

Nelson Francisco Favilla Ebecken

Department: System and Computing Engineering

This study aims to evaluate the text mining tools for categorizing character strings using real examples like amino acid sequences. For this purpose we developed a methodology capable of training a text classifier on some of sequences of proteins, and use this classifier trained to predict the category for the remaining unidentified sequences. That is, identify patterns of ordering of the amino acids for some sequences, which, considering its occurrence for several times, able to represent a category or class. Then, the classifier was designed on the original sequences to determine the similar regions of the sequences. From this point, it demonstrated effectiveness in addressing the prediction of the location of specific regions of these proteins to justify the possible alignments between the sequences, since classifiers successfully identify and categorize the sequences that share similar regions. The result is a sequence of learning classifier, which can be used to identify regions in proteins specific to the class. The experimental results agree with those predicted analytically.

Sumário

Capítulo 1 - Introdução	1
Capítulo 2 - Conceitos de Biologia Molecular	9
2.1 DNA – a molécula primária da vida	9
2.2 Células	10
2.3 Moléculas Biológicas	11
2.4 Proteínas	13
Capítulo 3 - Metodologia	16
3.1 Comparação de Sequências de Proteínas	16
3.1.1 Bancos de Dados de Sequências	18
3.1.2 Aplicações para Programas de Comparação de Sequências	19
3.1.2.1 Identificação de Semelhanças e Diferenças	19
3.1.2.2 Identificação de Funções Moleculares	19
3.1.2.3 Identificação de Anormalidades em Sequências	19
3.1.3 Programas para Comparação de Sequências	22
3.1.4 Matrizes de Substituição de Aminoácidos	28
3.1.4.1 Significância Estatística de Alinhamentos Locais de Sequências	29
3.1.4.2 Matrizes de Substituição Adequadas no Alinhamentos Locais de Sequências	30
3.1.4.3 Pontuações de Alinhamentos Locais usadas como Medidas de Informação	32
3.1.4.4 Seleção Adequada de uma Matriz de Substituição	33
3.1.5 Sensibilidade e Seletividade nas Comparações de Sequências	34
3.1.6 Eficiência dos Métodos de Comparação	35
3.2 Mineração de Textos	37
3.2.1 Análise dos Dados	40
3.2.2 Coleta de Dados	41
3.2.3 Pré-processamento	42
3.2.3.1 Geração de Vetores para Predição de Dados	44
3.2.4 Redução da Dimensionalidade	47
3.2.5 Preparação dos Dados	48
3.2.6 Similaridade entre termos	52
3.2.7 Indexação	54
3.2.7.1 Busca caractere-a caractere	55
3.2.8 Predição	56
3.2.8.1 Classificação de Documentos	58
3.2.8.2 Similaridade entre Documentos	59

3.2.9	Extração do Conhecimento	60
3.2.10	Categorização	61
3.2.10.1	Classificador Bayesiano	63
3.2.10.2	Classificador de Ranqueamento Linear	64
3.2.11	Avaliação dos Resultados	66
Capítulo 4	<i>Ferramenta de Mineração de Textos -Aíuri</i>	68
4.1	Funcionamento do Sistema Aíuri	69
4.2	Outras ferramentas de Mineração de Textos	73
4.2.1	Ferramentas de Domínio Público	74
4.3.1.1	Text Mine	74
4.3.1.2	TMSK - Text-Miner Software Kit	75
4.3.1.3	RIKTEXT - Rule Induction Kit for Text	76
4.3.1.4	Intext Software	76
4.3.2	Ferramentas Comerciais	77
4.3.2.1	TextSmart	77
4.3.2.2	STATISTICA Text Miner	78
4.3.2.3	SQL Server 2005	80
4.3.2.4	LexiQuest Categorize	80
Capítulo 5	<i>Testes e Avaliação dos Resultados</i>	82
5.1	Sequências de Proteínas utilizadas	82
5.2	Descrição do Problema	83
5.3	Proposta de Solução	83
5.3.1	Predição de Regiões de Similaridades	85
5.4	Descrição do Teste	87
5.4.1	Preparação dos Dados	88
5.4.2	Avaliação dos Resultados dos Categorizadores	89
5.4.2.1	Avaliação de Categorizador Bayesiano	91
5.4.2.2	Avaliação de Categorizador de Ranqueamento Linear	98
5.4.3	Avaliação dos Resultados da Ferramenta BLASTP	106
Capítulo 6	<i>Conclusões e Trabalhos Futuros</i>	111
6.1	Conclusões	112
6.2	Recomendações para Trabalhos Futuros	113
Capítulo 7	<i>Referências Bibliográficas</i>	115

Abreviaturas

ASCII	American Standard Code for Information Interchange
BLAST	Basic Local Alignment Search Tool
CluSTr	Clusters of SWISS-PROT and TrEMBL proteins
DNA	Deoxyribonucleic Acid (Ácido Desoxirribonucléico)
EBI	European Bioinformatics Institute
EI	Extração de Informações
EMBL	European Molecular Biology Laboratory
FASTA	Fast All
GUI	Graphic User Interface
HMM	Hidden Markov Models
HTML	HyperText Markup Language
idf	Inverse Document Frequency
IE	Information Extraction
KDD	Knowledge Discovery in Databases
KDT	Knowledge Discovery in Textual Databases
k-mer	n-tupla ou n-grama de ácido nucleico ou sequência de aminoácidos
ML	Machine Learning
MT	Mineração de Textos
MD	Mineração de Dados
Motif	Padrão de sequência de nucleotídeos em uma sequência de DNA ou de aminoácidos em uma proteína
NBRF	National Biomedical Research Foundation
NCBI	National Center for Biotechnology Information
NB	Naive Bayes
NLM	National Library of Medicine
NLP	Natural Language Processing
n-tupla	Quantidade requerida de resíduos consecutivos para alinhamento de sequências moleculares
PAM	Point Accepted Mutation
PDF	Portable Document Format
PIR	Protein Information Resource
PSM	Par de Segmentos de Comprimento Máximo
RI	Recuperação da Informação
RNA	Ribonucleic Acid
SQL	Structured Query Language
SVM	Support Vector Machine
TDM	Text Data Mining
tf	Term Frequency
tf-idf	Term Frequency–Inverse Document Frequency

VSM Vector Space Model
XML eXtensible Markup Language

Capítulo 1

Introdução

O advento de tecnologias mais rápidas no sequenciamento de DNA em meados de 1970, levou a uma explosão de informações na área biológica. Pesquisas sobre sequências moleculares tornaram-se o objetivo comum da área biomédica, permitindo o reconhecimento de *links* genéticos entre os mais diversos sistemas biológicos, promovendo inúmeros progressos na área da biologia molecular e se constituindo num método essencial para biólogos moleculares da atualidade. Ao mesmo tempo, que essas informações faziam as bases de sequências de DNA e proteínas crescerem a taxas muito elevadas, ocorria uma crescente demanda por recursos mais rápidos e eficientes na análise dessas sequências. Programas aplicativos para pesquisar similaridades entre sequências foram desenvolvidos, decrescendo sensivelmente o tempo requerido em comparações moleculares e transformando-se nas principais ferramentas dos biólogos na identificação de regiões codificadas e no acesso a informações sobre funções e estruturas de genes e proteínas.

A partir do ano de 1985, os cientistas David Lipman e Willian Pearson [43] deram início a um conjunto de programas aplicativos que mais tarde se constituiriam no principal recurso utilizado por biólogos e bioquímicos na comparação de sequências. O primeiro programa, denominado FASTP, baseava-se num algoritmo que buscava na

comparação de duas sequências, identificar as regiões que apresentassem similaridades, e a partir disso, atribuir-lhes uma pontuação (*score*) denotando acertos e erros de similaridades. Esse primeiro programa combinava uma rápida técnica de destacar as regiões num par de sequências que compartilham o mesmo grau de acertos de similaridades com um sistema de pontuação que considera além dos acertos, também as possibilidades de substituição entre aminoácidos/resíduos (pequenas moléculas unidas e formadoras de uma sequência de proteína) como similaridades válidas, utilizando como critério, matrizes de pontuação (ou substituição) que mensuram o grau de coincidências entre as cadeias de proteínas. Os programas de Lipman e Pearson foram um marco na utilização desses métodos de pesquisa, pois conseguiram reduzir acentuadamente o tempo de máquina na comparação de sequências, mas atualmente o desempenho desses programas não acompanha os anseios dos pesquisadores e profissionais da área.

Os algoritmos utilizados na comparação de sequências moleculares baseiam-se na busca de alinhamentos aproximados ótimos de cadeias (*strings*) [43]. Considerando uma cadeia (padrão) para consulta, uma ou mais cadeias de uma biblioteca, e uma variável k do tipo inteiro, o algoritmo busca todas as ocorrências da cadeia de consulta na biblioteca de cadeias, considerando no máximo a existência de k diferenças. Operações de edição (inserções, exclusões e substituições) corrigem as diferenças, convertendo uma cadeia na outra, ou seja, buscando encontrar o alinhamento máximo aproximado entre as cadeias. E quanto maior o número de similaridades entre estas cadeias, menor é o número de diferenças existentes entre ambas (distância de edição). Esses algoritmos permitem pesquisas eficientes em bases de dados de proteínas e DNA, focando apenas os grupos idênticos entre as duas sequências e, em decorrência disso, exigindo poucas comparações por consulta. Os métodos complementares utilizados na localização e análise das regiões de similaridades encontradas entre as sequências

comparadas são: construção de tabelas de comparação de posições e método da diagonal.

O surgimento de técnicas baseadas na Mineração de Textos [25] tornou viável a exploração de diversos tipos de textos em formato eletrônico que compõem o dia a dia de empresas e pessoas.

Com o crescimento contínuo do volume de dados eletrônicos disponíveis, técnicas de extração de conhecimento automáticas tornam-se cada vez mais necessárias para manipular essa gigantesca massa de dados. O principal objetivo das técnicas de mineração de textos é a manipulação de documentos em formato texto que se encontram de forma não-estruturada. De fato, as aplicações deste gênero fornecem uma nova dimensão das informações e que, se bem exploradas, podem se tornar um diferencial no domínio do negócio onde for aplicada. Desta maneira, um processo manual de avaliação e classificação de documentos torna-se inviável, na medida em que gera resultados baseados em uma visão limitada dos conteúdos dos documentos, ou demanda muito tempo para ser realizado.

Técnicas de aprendizado supervisionado sobre sequências tiveram muito sucesso na modelagem de proteínas. Alguns dos métodos mais utilizados são o Hidden Markov Models (HMMs) para modelar famílias de proteína e técnicas de redes neurais para a previsão de estrutura secundária.

Com o aumento da produção e disponibilidade de documentos em meio eletrônico, uma nova abordagem surgiu a partir da década de 90, baseada no aprendizado de máquina. Esse novo paradigma usa como base um conjunto de documentos pré-classificados em diversas categorias temáticas do domínio de interesse. A partir daí, um processo indutivo é aplicado para identificar as características que diferenciam as diversas categorias entre si.

Com base neste processo de indução, um modelo de classificação é construído, sendo usado para classificar documentos ainda não classificados. Para isso, são realizadas comparações entre as características presentes no novo documento com os padrões já mapeados dos documentos já conhecidos, e que participaram da construção do modelo. Este processo é denominado processo de aprendizagem supervisionada, uma vez que a construção do modelo de classificação é supervisionada por exemplos de documentos cujas categorias são conhecidas.

Compreensivelmente, os esforços para desenvolver ferramentas de *datamining* (mineração de dados) para minerar informações genéticas da literatura biomédica se intensificaram recentemente [62]

Como um primeiro passo, métodos automatizados de alta capacidade são necessários para validar rapidamente dados genéticos e identificar grupos de genes e proteínas relacionados funcionalmente, baseado na literatura publicada [34]. Uma vez que estes grupos de genes são identificados, métodos de *textmining* podem ser usados para extrair a natureza dos relacionamentos entre genes [72].

Recentemente uma nova classe de modelos que utilizam algoritmos de aprendizagem, tal como o algoritmo *Support Vector Machine* (SVM), foram aplicadas a modelagem de famílias de proteínas. Estes modelos incluem o espectro *kernel* e o *kernel* não alinhado, os quais já mostraram ser competitivos com métodos estado-da-arte na categorização de sequências de proteínas. Estes métodos representam sequências como coleções de *substrings* curtas de comprimento k ou k -mers. Uma propriedade desses classificadores é que podemos analisar a geração de modelos treinados por estes métodos e descobrir quais *k-mers* são os mais importantes para serem discriminados entre as categorias. Ao projetar estes *k-mers* sobre sequências de origem, pode-se descobrir quais regiões da proteína especificamente correspondem à categoria e

potencialmente apontam as regiões de maior similaridade da proteína. Em um recente estudo, demonstrou-se que alguns dos *k-mers* com os maiores pesos na classificação de um grupo de proteínas correspondem aos *motifs* conhecidos dessa família.

O objetivo deste trabalho foi treinar um classificador de textos sobre algumas das sequências de proteínas, e utilizar este classificador treinado para prever a categoria para as demais sequências não rotuladas. Em seguida, o categorizador é projetado sobre as sequências originais para determinar as regiões de similaridades das sequências.

A idéia foi combinar mineração de textos sobre a base de anotações com aprendizagem de texto para categorizar proteínas e determinar as regiões específicas similares entre as sequências de proteínas que representem uma categoria. Levando-se em consideração um pequeno conjunto exemplo de sequências não rotuladas (classificadas), o modelo definido aprende como as sequências estão relacionadas com os rótulos. O resultado da aprendizagem é uma sequência classificadora, que pode ser usada para identificar regiões nas proteínas específicas para a classe/categoria, ou seja, regiões que representem tal categoria devido a sua grande incidência nas proteínas pesquisadas.

A estrutura criada para comparar sequências de proteínas e classificá-las segundo um conjunto de exemplos de proteínas consiste das seguintes etapas: primeiramente, gera-se um conjunto de treinamento de proteínas rotuladas como exemplos positivos e a amostra de outras proteínas como exemplos negativos. Usando este conjunto de treinamento, treinam-se sequências classificadoras que serão utilizadas para prever sobre um banco de sequências àquelas correspondentes à categoria. A partir da projeção do conjunto classificador sobre outras sequências, é possível identificar quais regiões da proteína têm pesos positivos em relação à categoria do

conjunto exemplo de proteínas. E, dessa forma, considerá-las candidatas a regiões de relevante similaridade entre proteínas.

Em resumo, a entrada do método classificador utilizado é um conjunto de exemplos de proteínas e a saída é um classificador de sequências, o qual será utilizado para prever outros exemplos de proteínas que correspondam ao conjunto exemplo.

As previsões/predições para sequências desconhecidas recaem sobre os métodos utilizados para classificá-las. Após treinar um determinado número de exemplos, os métodos classificadores de texto podem prever com sucesso a classe correta de sequências não classificadas. O texto/sequência é classificado por meio do modelo *bag of words*, onde cada texto é mapeado em um vetor que contém a frequência de cada palavra. Como classificadores, são utilizados neste trabalho os seguintes métodos de categorização: bayesiana e linear. A categorização de textos é uma ferramenta utilizada para classificar automaticamente um conjunto de documentos em uma ou mais categorias preexistentes, tendo por finalidade recuperar a informação e identificar similaridades entre conjuntos de informações.

A ferramenta de mineração de textos utilizada no presente trabalho é o sistema Aûri [17], desenvolvido com base nos conceitos utilizados em aprendizado de máquina. A técnica de categorização automática de textos deste sistema consiste fundamentalmente em:

- (i) Disponibilizar um conjunto de documentos pré-classificados nas diversas categorias de interesse;
- (ii) Transformar a informação textual contida nos documentos para um formato que possa ser manipulado computacionalmente pelos algoritmos de classificação;

- (iii) Escolher os termos mais relevantes do conjunto de documentos, isto é, os que permitem melhor discriminação entre as categorias temáticas sob estudo;
- (iv) Definir os pesos para os termos dos documentos, de maneira a possibilitar uma maior discriminação entre as categorias consideradas;
- (v) Estimar o modelo de classificação;
- (vi) Avaliar a efetividade do modelo estimado.

O sistema Aíuri desenvolvido para um trabalho de Mestrado desta instituição, é segundo seu autor, Anddre Serpa [17], um sistema cooperativo acadêmico de alto desempenho - portal *web* - desenvolvido na linguagem de programação Java, que contém toda uma infra-estrutura para capacitá-lo a executar algoritmos em modo local ou em ambientes de *grids* computacionais. Estes algoritmos são capazes de determinar a qual classe um novo texto pertence baseado no conhecimento obtido de textos anteriores.

Uma segunda ferramenta foi desenvolvida durante este trabalho para uma adequada preparação dos dados dos arquivos de proteínas. Esta ferramenta consiste em particionar as longas cadeias de proteínas em fragmentos de comprimentos variados dentro dos próprios arquivos, os quais são utilizados como dados de entrada para a ferramenta mineradora após conversão destes para formatos adequados (.txt e .xml) pela própria.

A ferramenta de bioinformática BLAST (Ferramenta para Busca de Alinhamento Local) amplamente utilizada, cuja função é encontrar regiões de similaridade entre sequências e classificá-las, foi apresentada neste trabalho como modelo de ferramenta que, por possuir aspectos funcionais similares às ferramentas de categorização de textos (e cadeias de caracteres), pode inspirar futuros trabalhos que aperfeiçoem os métodos utilizados nesta tese. O programa BLAST, além de calcular a

significância estatística dos alinhamentos perfeitos e utilizar um eficiente sistema de pontuação, pode também ser usado para classificar os relacionamentos funcionais e evolutivos entre sequências tão bem quanto identificar membros de uma família genética.

Estrutura da tese

Este trabalho é estruturado como se segue. No capítulo 2 apresentam-se fundamentos da biologia molecular. No capítulo 3 são apresentadas as técnicas, conceitos e algoritmos utilizados nas atividades de comparação de sequências moleculares e mineração de textos. No capítulo 4 são apresentadas as ferramentas comerciais e acadêmicas utilizadas na mineração de texto. O capítulo 5 é dedicado à apresentação de resultados dos testes do ambiente desenvolvido. No capítulo 6 são delineadas as conclusões e perspectivas de trabalhos futuros. E finalmente, no capítulo 7 as referências bibliográficas são apresentadas.

Capítulo 2

Conceitos de Biologia Molecular

Neste capítulo é feita uma abordagem clara e objetiva sobre os principais termos técnicos de biologia molecular, os quais são mencionados no decorrer de toda esta dissertação de tese.

2.1 DNA – a molécula primária da vida

A molécula de DNA carrega na sua estrutura a informação hereditária que determina a estrutura das proteínas. *O DNA é a molécula primária da vida.* As instruções que comandam as células no crescimento e na divisão (suas funções básicas), são codificadas pelo DNA. São as mensagens codificadas que levam a diferenciação das características dos ovos fertilizados a uma multidão de células importantes e necessárias ao funcionamento dos seres vivos. O DNA é a base do processo evolucionário das diversas espécies de organismos vivos. E a extraordinária capacidade que as moléculas modificadas de DNA têm de dar origem a novas formas de vida, mais aptas à sobrevivência que seus progenitores, possibilitou o aparecimento da nossa própria espécie [58].

O DNA é descrito como a molécula que carrega a informação genética do ser vivo. Somente em 1953 revelou-se que a estrutura do DNA se constituía de uma

dupla hélice [69]. É do DNA que emanam os comandos que regulam a natureza de cada tipo de molécula celular. Assim, muitas das atenções têm-se voltado, cada vez mais, para a descoberta de informações contidas no DNA. Pois, revelando-se de forma exata o esquema genético humano, um grande passo será dado no entendimento da complexidade das reações químicas interconectadas que levam ovos fertilizados a se desenvolverem em organismos multicelulares altamente complexos.

Somente examinando detalhadamente (sequenciando) o DNA, podemos identificar os comandos (as funções celulares) que este pode gerar.

2.2 Células

As células são os blocos construtivos de todos os seres vivos. Geralmente são muito pequenas, com diâmetros inferiores a 1mm, tornando-se invisíveis a olho nu. Na mais simples das células – a bactéria – uma parede celular recobre uma fina membrana externa contendo ácidos graxos na sua constituição (membrana plasmática), que circunda uma região interna não estruturada superficialmente. Dentro dessa região está localizado o DNA bacteriano, que carrega a informação genética. A membrana plasmática é efetivamente impermeável (exceto para algumas moléculas nutrientes e íons selecionados) de tal maneira que o conteúdo fique retido no seu interior e não extravase. A integridade dessa membrana externa é, portanto, essencial à vida da célula.

Praticamente todas as células, com exceção da célula bacteriana, têm sua massa celular interna compartimentada por dois sistemas de membranas: um que circunda um corpo esférico, denominado núcleo, e outro circundando o citoplasma. No núcleo está localizado o DNA celular, na forma de cilindros enrolados denominados cromossomos. As células que contêm núcleo são chamadas

eucarióticas, enquanto que as células sem núcleo, como as bactérias e outros organismos relacionados (algas verdes-azuis), são conhecidas como procarióticas.

As células são pequenas fábricas de moléculas, ou seja, têm a capacidade de crescer e dividir-se, produzindo células-filhas capazes de formar novas moléculas, de se replicarem. Para executarem essas funções, as células precisam quimicamente ser muito sofisticadas. Assim, as células crescem por serem capazes como pequenas fábricas, de absorver moléculas precursoras simples, como glicose e dióxido de carbono, e de transformá-las, de alguma forma, em diversas moléculas contendo carbono, que são essenciais ao funcionamento celular. Durante seu crescimento e sua divisão, as células requerem, também, fontes externas de energia, a fim de assegurar que as reações químicas ocorram em direção ao processo biossintético. Para a maioria das células, a energia necessária a sua manutenção e crescimento é obtida da quebra das moléculas nutrientes, enquanto que aquelas capazes de promover a fotossíntese usam diretamente a energia solar.

2.3 Moléculas Biológicas

As moléculas que ocorrem nas células podem ser divididas em duas classes distintas de acordo com o seu tamanho. Uma classe é denominada *pequenas moléculas* (açúcares, *aminoácidos* e ácidos graxos). A segunda classe é constituída pelas *macromoléculas*, onde destacamos as *proteínas* e os ácidos nucleicos. As macromoléculas são moléculas *poliméricas* (compostas) formadas pela reunião de tipos específicos de pequenas moléculas arrumadas em longas cadeias. As moléculas poliméricas contêm um grande número de subunidades *monoméricas*, onde as primeiras são centenas ou milhares de vezes maiores que as pequenas moléculas.

Assim, considerar somente o número e o tamanho de suas moléculas é suficiente para tornar a mais simples das células em uma entidade extremamente complexa do ponto de vista químico. Entretanto, a singularidade química das células repousa menos na complexidade inerente às suas moléculas individuais do que na natureza das reações químicas que transformam uma molécula em outra.

Dessas reações as mais importantes são aquelas que: 1) levam à quebra das moléculas nutrientes em unidades moleculares menores que podem ser reassociadas para a formação de componentes moleculares vitais às células; 2) aproveitam a maior parte da energia liberada pela quebra dos alimentos (ou absorção da luz; 3) sintetizam pequenas moléculas precursoras que serão, posteriormente, utilizadas para a síntese de macromoléculas; 4) associam as subunidades monoméricas em macromoléculas altamente ordenadas.

Tabela 1 – Os Vinte Aminoácidos Construtores das Proteínas

<i>Aminoácido</i>	<i>Abreviatura(três letras)</i>	<i>Código (uma letra)</i>
Icina	Gli	G
Alanina	Ala	A
Valina	Val	V
Isoleucina	Iso	I
Leucina	Leu	L
Serina	Ser	S
Treonina	Tre	T
Prolina	Pro	P
Ácido Aspártico	Asp	D
Ácido Glutâmico	Glu	E
Lisina	Lis	K
Arginina	Arg	R
Asparagina	Asn	N
Glutamina	Gln	Q
Cisteína	Cis	C
Metionina	Met	M
Triptofano	Trp	W
Fenilalanina	Fen	F
Tirosina	Tir	Y
Histidina	His	H

2.4 Proteínas

As proteínas ficaram conhecidas como sendo moléculas poliméricas formadas por aminoácidos ligados uns aos outros, formando cadeias polipeptídicas. A grande maioria das proteínas é constituída de uma mistura dos mesmos vinte aminoácidos, sendo que o percentual de um dado aminoácido varia de uma proteína para outra. Em 1953, foi observado que cada cadeia é então caracterizada por uma determinada sequência de aminoácidos [61].

Muitas proteínas contêm somente uma sequência de aminoácidos ao longo de sua cadeia polipeptídica. Entretanto, existem muitas proteínas que são formadas pela associação de cadeias contendo diferentes sequências. Em geral, o tamanho de diferentes proteínas varia muito; as suas cadeias componentes podem conter de 20 até mais de quatro mil aminoácidos.

Por exemplo, a proteína transportadora de oxigênio, a hemoglobina, é formada pela associação de quatro cadeias polipeptídicas, duas com uma sequência específica α e duas contendo a sequência β .

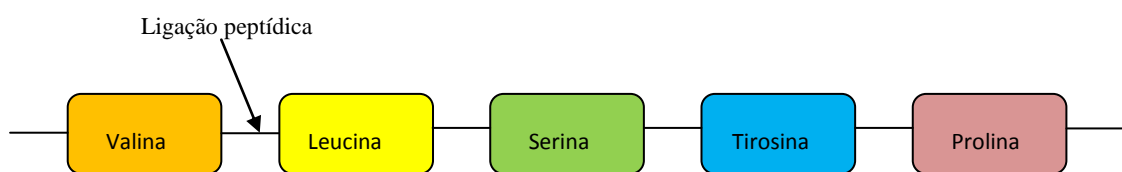


Figura 1 – Os aminoácidos são blocos construtores de uma proteína

Embora muitas das proteínas celulares sejam enzimas, existem proteínas com função estrutural, também. Elas auxiliam, por exemplo, na formação do arcabouço da membrana externa e da membrana nuclear. Proteínas, tais como o colágeno fibroso, fazem parte do tecido conectivo entre as células. Outras proteínas como a actina e a

tropomiosina, são constituintes da fibra muscular. E outras, dentre elas a calmodulina, funcionam como ligantes de íons Ca^{2+} que regula a atividade de outras enzimas. Finalmente, existem proteínas que funcionam extracelularmente, tais como o hormônio insulina e os anticorpos, esses últimos responsáveis pela manutenção da resposta imune do nosso organismo.

A função de uma proteína depende de um dobramento preciso de sua cadeia polipeptídica (estrutura tridimensional). Enquanto algumas proteínas se dobram, formando longos cilindros, uma grande parte das enzimas contém cavidades formadas por cadeias laterais de aminoácidos apropriados. A sequência específica de uma dada enzima é muito importante. Se aminoácidos não apropriados estiverem presentes, a cadeia polipeptídica poderá não sofrer o dobramento adequado para exercer a sua atividade catalítica. Pelo fato de possuir vinte aminoácidos disponíveis, cada um deles com forma e propriedades químicas particulares e capazes de formar cadeias polipeptídicas de diferentes tamanhos, a célula tem uma capacidade enorme de desenvolver novas proteínas que possuam cavidades catalíticas com formas complementares a qualquer substrato em potencial.

As enzimas são capazes de dirigir as reações químicas que catalisam em direções que promovam ordenadamente o crescimento e a divisão celular. A direção das reações químicas é, de fato, determinada pela lei da termodinâmica. Nas células em crescimento, a polimerização de aminoácidos em polipeptídeos é mais favorecida em relação à quebra das cadeias polipeptídicas em seus aminoácidos monoméricos, mesmo sendo essa última energeticamente mais vantajosa.

As enzimas não podem determinar a ordem dos aminoácidos nas cadeias polipeptídicas. Embora as enzimas determinem a especificidade das reações químicas

entre pequenas moléculas, não existe uma maneira pela qual elas possam ser usadas para determinar a ordem dos aminoácidos em milhares de diferentes proteínas celulares. Uma cadeia polipeptídica de tamanho médio contém várias centenas de aminoácidos numa sequência única. Se a ordem dos aminoácidos na cadeia fosse determinada por enzimas, dever-se-ia ter um número exorbitante de enzimas, cada uma capaz de reconhecer um grande número de aminoácidos contíguos. Assim, todas essas hipotéticas enzimas reconhecedoras de uma sequência de aminoácidos deveriam ser colocadas juntas numa série de diferentes enzimas reconhecedoras de sequência e assim por diante. Esse tipo de esquema é, obviamente, impossível de ser operacionalizado e nos leva à conclusão de que as células precisam conter moléculas específicas “carregadoras de informação”, análogas, talvez, aos moldes de um escultor ou às placas-mestras de um litógrafo. Tais moléculas precisam codificar uma informação ordenada que possa ser usada para selecionar o aminoácido correto no curso da síntese do polipeptídeo e precisam, de alguma maneira, ser capazes de sintetizar novas cópias de si mesmas (autoduplicação) de tal forma que, quando a célula em crescimento gerar duas células-filhas, cada uma delas possua cópias do molde original.

A função de uma proteína depende diretamente das cadeias de aminoácidos constituintes de uma proteína. A sequência específica de uma proteína é importante, pelo fato de possuir vinte aminoácidos, cada um deles com forma e propriedades químicas particulares, distribuídos em percentuais variados nas sequências, de forma a gerar estruturas com conformações diversas.

Capítulo 3

Metodologia

Para o desenvolvimento deste trabalho, foram estudadas as áreas de comparação e alinhamento de sequências moleculares de proteínas (buscando revelar as regiões funcionais importantes) e de mineração de texto (trabalhando com a similaridade entre termos e cadeias de caracteres).

3.1 Comparação de Sequências de Proteínas

Vários algoritmos foram desenvolvidos e implementados provendo interfaces gráficas amigáveis (GUI) para os bancos de dados existentes. Estes algoritmos mais rápidos utilizados na comparação de sequências de DNA e sequências de proteínas vêm, ao longo de quase duas décadas, decrescendo sensivelmente a quantidade de tempo requerida na comparação de uma nova (não identificada) sequência com bases de dados sobre sequências de DNA e proteína. Estes métodos chegaram numa época em que as bibliotecas sobre essas sequências cresciam a taxas elevadas, devendo-se isso as mais eficientes técnicas de clonagem e aos mais produtivos procedimentos de sequenciamento¹. As pesquisas de sequências levaram a descoberta de diversas novas

¹ Procedimento que envolve métodos de pesquisa de sequências moleculares genéticas ou de proteínas, em busca de informações sobre novas sequências.

famílias de proteínas. Geralmente uma consulta a uma base de dados conduz inicialmente ao que poderíamos chamar de uma amostra inicial de uma nova proteína sequenciada. Não obstante, ainda é necessária uma cuidadosa análise dos resultados e uma seleção mais fina nas pesquisas aos diversos bancos de sequências existentes. A bioinformática pode então conduzir a uma melhor compreensão da vida e das causas moleculares de certas doenças.

Comparando-se sequências moleculares (DNA e proteína), o biólogo pode realizar diversas descobertas, tais como: antepassados de organismos; árvores filogenéticas²; estruturas de proteínas; funções de proteínas; e estruturas genéticas (DNA).

A comparação de sequências é uma ferramenta muito poderosa na biologia molecular, genética e química protéica. Frequentemente é desconhecido que tipo de proteínas uma nova sequência de DNA codifica. Na comparação de uma nova sequência codificada com outras sequências já conhecidas, há probabilidade de se encontrar uma sequência similar. Geralmente se conhecem certas funções celulares das proteínas dos bancos de dados. Considerando-se que sequências semelhantes implicam em funções semelhantes, observa-se que o atual conhecimento sobre novas sequências é muito mais amplo [21].

Determinadas classes de proteínas (referências) frequentemente exibem alguns aminoácidos que sempre ocorrem nas mesmas posições em sequências de proteínas. Com o uso destes “padrões”, existe a possibilidade de se obter informações sobre a atividade de uma proteína, da qual apenas o gene (DNA) é conhecido. A avaliação de tais padrões produz informações sobre a arquitetura de proteínas.

² Árvores filogenéticas são árvores genealógicas que são construídas a partir de informações obtidas na comparação das sequências de aminoácidos de uma proteína.

3.1.1 Bancos de Dados de Sequências

Nas últimas décadas foram criados alguns bancos de dados para armazenar o grande número de sequ de DNA, RNA e proteínas e com isso também foram desenvolvidas técnicas computacionais para permitir uma procura rápida nesses bancos de dados [48]. É importante frisar que esses bancos de dados do genoma, são bancos diferentes dos usados na área de informática que são, por exemplo, do tipo relacional, como por exemplo o *Interbase*. No caso do genoma as informações são armazenadas e atualizadas pelos biólogos em arquivos do tipo texto e que por eles são chamados de banco de dados [48].

Principais bancos de dados representativos de seqüenciamentos: *GenBank*: é um banco de dados público de sequências de nucleotídeos e proteínas com informações biológicas, produzido e distribuído pelo *National Center for Biotechnology Information* (NCBI), uma divisão do *National Library of Medicine* (NLM), localizado no campus da *US National Institutes of Health* (NIH); PIR-International: *Protein Information Resource – International* é um banco de dados de sequências de proteínas que foi iniciado no *National Biomedical Research Foundation* (NBRF) no início do ano de 1960; *Swiss-Prot* é um banco de proteínas mantido pelo *Swiss Institute of Bioinformatics* (SIB) em colaboração com o *European Bioinformatics Institute* (EBI).

Além dos dados da sequência, o banco de dados contém informações que concentram: (1) o nome e a classificação da proteína e do organismo onde esta ocorre; (2) referências à literatura principal, incluindo informações sobre a determinação da sequência; (3) as características funcionais e gerais da proteína; e (4) locais/sítios de interesse biológico dentro da sequência. Os registros do banco de dados possuem referências cruzadas para os bancos de dados originais. Conceitualmente o banco de

dados consiste em três componentes principais: literatura, sequência, e anotações da proteína.

3.1.2 Aplicações para Programas de Comparação de Sequências

3.1.2.1 Identificação de Semelhanças e Diferenças

Com os programas de comparação de sequências é possível encontrar a proteína ou o DNA correspondente a outras sequências das bases de sequências moleculares. Isto pode ser obtido por um dado alinhamento de pares (pairwise³) de resíduos ocorrido entre sequências, o que pode fornecer pistas para se determinar a possível função da molécula.

3.1.2.2 Identificação de Funções Moleculares

Soluções computacionais para comparar sequências contra uma vasta quantidade de sequências já analisadas podem ajudar a fazer classificações concernentes as suas semelhanças e diferenças e permitir hipóteses sobre suas funções. Também poderiam revelar que duas moléculas compartilham a mesma função porque têm os mesmos domínios embora ocorram em compartimentos diferentes da célula (citoplasma ou núcleo) ou, até mesmo, em organismos diferentes.

3.1.2.3 Identificação de Anormalidades em Sequências

Proteínas com as mesmas funções, mas de origens diferentes, normalmente têm variações nas sequências de aminoácidos, mas continuam funcionais. Estas variações na sequência dos blocos construtores são causadas pelas mutações na sequência de DNA correspondente, isto é, o nucleotídeo dito "normal" é substituído por um diferente. Enquanto estas mutações normalmente não são prejudiciais à função de uma molécula, também há mutações que tornam uma molécula inativa. A identificação de uma

³ Alinhamento Pairwise, é um alinhamento ocorrido entre pares de resíduos, o qual recebe uma pontuação ótima (i.e. pares com máxima semelhança).

sequência mutada, ou seja, uma sequência de aminoácidos com pelo menos um aminoácido diferente do original poderia fornecer evidências de que um certo aminoácido é responsável por alguma anormalidade da função molecular.

Um bom exemplo para isto é a anemia falciforme que foi a primeira doença de origem molecular identificada. Esta doença é causada pela substituição do aminoácido ácido glutâmico pelo valina na posição 6 (seis) da cadeia. o que conduz a uma má formação da proteína hemoglobina.

Outro exemplo muito proeminente é a diabetes mellitus, uma doença causada pela inabilidade do corpo para produzir insulina suficiente. Em contraste com o exemplo anterior, a diabetes não é uma doença que seja causada pelo mau funcionamento de uma proteína. A cura atualmente mais efetiva para esta doença é prover o indivíduo com insulina. Antigamente esta insulina era coletada dos porcos, mas recentemente foi possível produzir insulina humana em bactéria, a qual pode ser produzida em qualquer quantidade e qualidade.

A Figura 2 mostra a razão porque o porco, ou em geral, a insulina derivada do animal pode ser usada para tratar a diabetes: Como pode ser visto nesta figura, as sequências de aminoácidos das insulinas animais são bem semelhantes a da forma humana. Aminoácidos (simbolizados pelo código de uma letra) que se alinham/emparelham corretamente estão exibidos entre as sequências e marcados por hífen azuis. Os que não se alinham estão marcados com barras vermelhas.

A identidade da sequência é de 90% para coelhos; 87% para porcos, e 83% para vacas.

```

sp|P01311|INS RABIT INSULIN PRECURSOR.
      Length = 110

Score = 424 (199.8 bits), Expect = 3.4e-56, P = 3.4e-56
Identities = 78/86 (90%), Positives = 81/86 (94%)

Query:   25 FVNQHLCGSHLVEALYLVCGERGFFYTPKTRREAEDLQVQVELGGGPGAGSLQPLALEG 84
         FVNQHLCGSHLVEALYLVCGERGFFYTPK+RRE E+LQVGQ ELGGGPGAG LQP ALE
Sbjct:   25 FVNQHLCGSHLVEALYLVCGERGFFYTPKSRREVEELQVGQAELGGGPGAGGLQPSALEL 84

Query:   85 SLQKRGIVEQCCTSICSLYQLENYCN 110      Human
         +LQKRGIVEQCCTSICSLYQLENYCN
Sbjct:   85 ALQKRGIVEQCCTSICSLYQLENYCN 110      Rabbit

```

```

sp|P01315|INS PIG PROINSULIN.
      Length = 84

Score = 229 (107.9 bits), Expect = 1.1e-52, Sum P(2) = 1.1e-52
Identities = 43/49 (87%), Positives = 44/49 (89%)

Query:   25 FVNQHLCGSHLVEALYLVCGERGFFYTPKTRREAEDLQVQVELGGGPG 73      Human
         FVNQHLCGSHLVEALYLVCGERGFFYTPK RREA E+ Q G VELGGG G
Sbjct:   1 FVNQHLCGSHLVEALYLVCGERGFFYTPKARREAENPQAGAVELGGGLG 49      Pig

```

```

sp|P01317|INS BOVIN INSULIN PRECURSOR.
      Length = 105

Score = 244 (115.0 bits), Expect = 1.1e-49, Sum P(2) = 1.1e-49
Identities = 45/54 (83%), Positives = 47/54 (87%)

Query:   25 FVNQHLCGSHLVEALYLVCGERGFFYTPKTRREAEDLQVQVELGGGPGAGSLQ 78      Human
         FVNQHLCGSHLVEALYLVCGERGFFYTPK RRE E QVG +EL GGPGAG L+
Sbjct:   25 FVNQHLCGSHLVEALYLVCGERGFFYTPKARREVEGPPQVGALELAGGPGAGGLE 78      Cow

```

Figura 2 - Comparação da sequênciada insulina humana com as insulinas dos coelho, porco, e vaca, respectivamente. As imagens são derivadas de uma pesquisa com o programa BLAST contra o banco de dados SWISS-PROT.

As sequências são representadas por um código de uma letra para aminoácidos, assim a mesma letra em duas linhas é equivalente a um alinhamento completo. São introduzidos espaços (*gaps*⁴) em uma sequência para estender uma determinada sequência, de forma a obter-se um melhor alinhamento total. Os números próximos às sequências individuais indicam a localização dos resíduos/aminoácidos individuais na cadeia.

⁴ Espaços interpostos entre alinhamentos de sequências, de forma a dar mais sensibilidade à comparação.

Em regiões altamente conservadas (sem diferenças) de uma proteína são permitidas mutações conservadoras que não corrompam a função da proteína. Nestas mutações conservadoras, a substituição de um aminoácido por um outro, que tenha propriedades físicoquímicas comparáveis, pode ser detectada por um programa de comparação, o qual pode "decidir" em que grau uma sequência é semelhante a outra.

3.1.3 Programas para Comparação de Sequências

Os programas para comparação de sequências geralmente trabalham apenas com três entradas, a sequência a ser comparada, um banco (arquivo) de sequências ou uma segunda sequência para comparação, e o valor do parâmetro *ntupla* (quantidade requerida de resíduos consecutivos para alinhamento) de comparação (*ntupla*=1 para comparações unárias; *ntupla*=2 para comparações binárias; etc.). Os programas comparam a primeira sequência com todas as outras sequências do arquivo, fornecendo a melhor pontuação por similaridade e alinhamento para cada comparação de par de resíduos. Estes programas calculam um *score* (pontuação) de similaridade local, ou seja, a melhor região de similaridade encontrada na comparação entre duas sequências.

A Tabela 2 mostra o resultado da comparação entre a proteína bovina AMP-kinase dependente e as 2677 sequências de proteínas da base NBRF⁵ [43], onde as 20 sequências mais similares e suas pontuações (*scores*) são apresentadas.

Alguns programas informam apenas a pontuação de similaridade para o melhor alinhamento de pares de resíduos ocorrido entre duas sequências. No caso de proteínas que possuam repetidos grupos de resíduos (aminoácidos) em suas constituições, podem existir vários acertos (alinhamentos perfeitos) com pontuações

⁵ National Biomedical Research Foundation

indicadoras de grande similaridade. Outros programas informam também as múltiplas regiões de similaridades (grandes ou pequenas) e tratam-nas como alinhamentos decorrentes da comparação entre sequências.

Tabela 2 – Sequências de proteínas similares a sequência bovine cyclic AMP-dependent kinase

<i>Base NBRF</i> <i>Proteína</i>	<i>Definição</i>	<i>Score</i>	
		<i>Inicial</i>	<i>Otimizado</i>
OKBO2C	Bovine cyclic AMP-dependent kinase	1810	1810
TVYUH	Avian erythroblastosis virus kinase-related protein	96	179
GNMVRR	Feline sarcoma virus	90	191
TVBY8	Yeast cell division control protein	88	224
TVFV-R	Rous sarcoma virus kinase-related protein	86	217
TVCHS	Chicken kinase-related protein	86	212
TVFV60	Rous sarcoma virus (avian sarcoma virus)	86	216
GNFVG9	Avian sarcoma virus p90	86	203
TVRT-M	Rat kinase-related protein	85	112
GNVWGM	Abelson murine leukemia virus	83	237
GNVVF	Fujinami sarcoma virus	80	190
TVHU-T	Human probable kinase-related protein	78	133
TVMV-M	Moloney murine sarcoma virus	76	111
TVMV1M	Moloney murine sarcoma virus	76	110
TVMS-M	Mouse kinase-related protein	76	108
GNMVCS	Feline sarcoma virus	62	161
GNMVGC	Feline sarcoma virus	62	163
QQECUC	E. coli unc hypothetical protein	51	51
CYBOB	Bovine beta crystallin Bp chain	51	54
HBGTF	Goat and sheep hemoglobin beta fetal chain	49	49

A maioria dos algoritmos de consultas de semelhanças entre sequências de DNA e proteínas baseavam-se no princípio de comparação de cada nucleotídeo ou aminoácido de uma sequência com todos os resíduos de uma segunda sequência. Por meio destes algoritmos a comparação de uma cadeia de 200 aminoácidos com uma biblioteca de proteínas de 500.000 resíduos demandaria aproximadamente 10^8 comparações. A atual geração [73] descreve um algoritmo que permite pesquisas rápidas em bases de dados de proteínas e ácidos nucléicos focando apenas os grupos idênticos entre as duas sequências e em decorrência disso exigindo poucas comparações por consulta [43][53].

No algoritmo original, resíduos alinhados ou grupos de resíduos alinhados são rapidamente localizados através da construção de uma tabela verdade. Uma

tabela com o posicionamento de cada resíduo/aminoácido é construída para cada sequência (sequência 1) da base de sequências a ser comparada com a sequência de consulta (sequência 2), considerando pesquisas por resíduo (ntupla=1) ou por pares de resíduos (ntupla=2) . A pesquisa por mais resíduos, valores entre 4 a 6, é indicada na comparação de sequências de DNA. A comparação visa encontrar a ocorrência de aminoácidos da sequência 2 (sequência de consulta) na tabela de posições da sequência 1. Assim, de acordo com a Tabela 3 do exemplo, o resíduo S da Segunda sequência ocorre no resíduo S (posição 3) da primeira sequência, e W da segunda sequência aparece nas posições 3 e 6 da primeira.

Tabela 3 – Exemplo de uma comparação de sequências

Sequência	Posição						
	1	2	3	4	5	6	7
1	F	L	W	R	T	W	S
2	S	W	K	T	W	T	

Em conjunto com a tabela verdade, o “método da diagonal” [73] foi desenvolvido para localizar todas as regiões de similaridade entre duas sequências, contabilizando os alinhamentos de resíduos e penalizando os não-alinhamentos distribuídos pelas regiões. Este método identifica regiões de uma diagonal que possua a mais alta densidade em alinhamentos. O termo diagonal; refere-se a linha diagonal traçada entre duas sequências comparadas, a diagonal denota o alinhamento/acerto entre as sequências sem qualquer ocorrência de intervalos. Para cada resíduo (ou pares de resíduos) encontrado através da tabela de busca, a diferença entre as posições de alinhamento das duas sequências é calculada. Algumas diferenças serão equivalentes para acertos de diferentes resíduos, as quais podem ser

simultaneamente alinhadas sem a introdução de intervalos. Uma linha diagonal plotada numa matriz de pontos é composta por acertos que têm iguais valores de diferenças.

No exemplo, o resíduo S na posição 1 da sequência 2 (posição corrente na sequência de consulta) casa com S da sequência 1 (posição corrente na biblioteca de sequências) na diferença de $7 - 1 = 6$, ou seja, na diagonal 6 (Figura 3); o W na posição 2 da sequência 2 ocorre nas diagonais $(3 - 2)1$ e $(6 - 2)4$; o T na posição 4 da sequência 2 ocorre na diagonal 1; W na posição 5 da sequência 2 nas diagonais 1 e -2; e o próximo T na diagonal -1. Compartilhando a mesma diagonal, a diagonal 1, temos três acertos (casamentos) de identificação (W(2), W(5) e T(4)), enquanto que para as outras diagonais temos 1 ou zero alinhamento. O método diagonal compara as duas sequências varrendo sequência 2 do início ao fim; a pontuação de uma diagonal é incrementada a cada acerto e decrementada a cada erro. As diagonais com pontuações mais altas representam as regiões locais de similaridades entre duas sequências. No caso do exemplo (Figura 3) analisado seriam 5 (cinco) as regiões locais com pontuações (*scores*) mais significativos. Após este estágio, o algoritmo desenvolvido por Wilbur e Lipman [73] calcula um ponto de similaridade final, permitindo *inserções e exclusões*⁶, baseadas em todos os acertos ou em blocos de acertos localizados dentro de uma distância preestabelecida de quaisquer das regiões locais de similaridade.

⁶ Numa comparação de sequências, as diferenças existentes entre ambas podem ser corrigidas (distância de correção) através de substituições, inserções e exclusões, convertendo uma sequênciana outra. Existem três tipos de diferenças:

- (1) um resíduo da sequência de consulta corresponde a um diferente resíduo da seqüenciada biblioteca de sequências (uma substituição);
- (2) um resíduo da sequência de consulta corresponde a nenhum resíduo da seqüenciada biblioteca (uma inserção);
- (3) um resíduo da sequência da biblioteca corresponde a nenhum resíduo da seqüenciade consulta (uma exclusão).

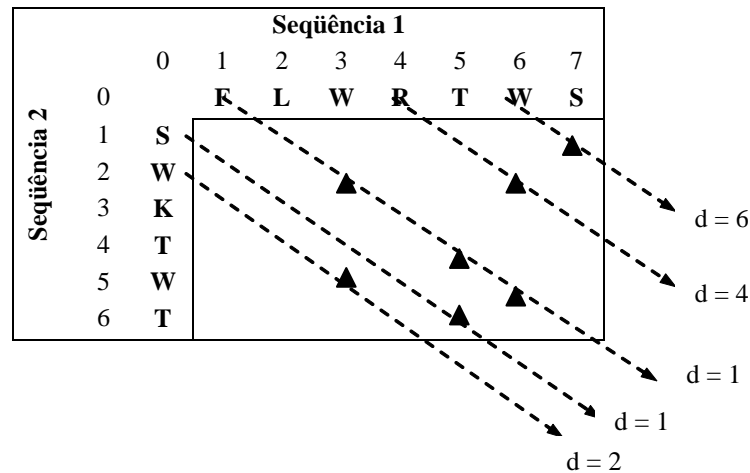


Figura 3 - Representação das diagonais e regiões locais de similaridade entre duas sequências

O algoritmo visa principalmente modificar o método da diagonal, aumentando drasticamente a eficiência e a *sensibilidade nas comparações*⁷ entre cadeias de aminoácidos. Este algoritmo busca localizar as posições inicial e final em ambas as sequências das cinco regiões de maior similaridade encontradas pelo método da diagonal.

No exemplo, a melhor região de similaridade está na diagonal 1 e se estende dos resíduos 3 a 6 da sequência 1.

As cinco regiões com as mais altas pontuações sofrem uma recontagem dessas pontuações comparando-se a similaridade dos pares de aminoácidos (substituições e acertos), usando uma *matriz de substituição*⁸ de aminoácidos, a

⁷ Na comparação de sequências, existe uma estreita relação entre a **sensibilidade** (a habilidade de identificar sequências de origens distantes/remotas, mas correlacionadas) e a **seletividade** (a forma de evitar os falsos pontos positivos, ou seja, sequências não correlacionadas, mas com altos pontos de similaridade).

⁸ Uma matriz de substituição é um critério utilizado para medir o grau de coincidência do alinhamento de resíduos obtido entre duas sequências. Uma matriz auxilia na verificação da probabilidade de um certo resíduo ocorrer com certa frequência num alinhamento entre sequências, atribuindo pontos para esta ocorrência. Existem diversos modelos de matrizes, cada um adequado a um tipo de alinhamento.

matriz PAM250⁹. Aminoácidos idênticos alinhados (como cisteína e triptofano), recebem pontos mais altos do que alinhamentos mais comuns entre aminoácidos (tais como serina e alanina), e substituições que ocorrem muito freqüentemente, como a metionina por leucina, também recebem *pontos* positivos enquanto substituições inadequadas ou menos freqüentes (tais como cisteína por triptofano) recebem pontos negativos. No exemplo, os resíduos alinhados W e T obtêm pontuações 17 e 3, respectivamente, e a substituição de K por R recebe uma pontuação de 3, para uma total de pontos de 40.

A pontuação da subsequência que apresenta o melhor alinhamento, baseado numa matriz de substituição, é utilizada como a pontuação mais representativa da similaridade entre as duas sequências (a pontuação inicial). Quando se realiza uma pesquisa no banco de sequências, uma pontuação inicial é calculada entre a sequência de consulta e cada uma das sequências do banco. As sequências são ordenadas pela sua pontuação inicial. Além disso, uma pontuação otimizada, a qual permite inserções e exclusões de resíduos, também é computada para as sequências que apresentam as maiores pontuações iniciais de similaridade. O algoritmo realiza uma otimização local, onde porções não similares encontradas externamente à região de alinhamento ótimo não afetam o valor final da pontuação de alinhamento.

Os programas comparam a primeira sequência (sequência de consulta) com todas as sequências da biblioteca, informando a melhor pontuação de similaridade para cada alinhamento entre resíduos das sequências. Estes programas calculam também uma pontuação de similaridade local, ou seja, a melhor região de

⁹ A matriz PAM250 foi inicialmente proposta [19] para alinhamentos globais de sequências homólogas, mas vem sendo amplamente utilizada em alinhamentos locais também [42] [43]. Essa matriz derivou-se da comparação de várias proteínas relacionadas e determina a probabilidade de um dado aminoácido ser substituído por outros aminoácidos num certo espaço de tempo. Estudos empíricos mostram que a matriz PAM250 permite encontrar os relacionamentos mais remotos entre sequências.

similaridade entre duas sequências. A pontuação da região local não é afetada por porções das sequências pobremente alinhadas fora da melhor região.

Estes algoritmos informam apenas a pontuação de similaridade para o melhor alinhamento de pares entre duas sequências. No caso de proteínas com domínios repetidos, podem existir vários alinhamentos com altas pontuações por similaridade, e que são de interesse biológico. Múltiplas regiões de similaridade podem ser representadas como alinhamentos entre sequências, mas procurando fazer distinção entre regiões de fortes semelhanças e regiões com pontuações baixas por similaridade.

3.1.4 Matrizes de Substituição de Aminoácidos

Alinhamentos locais são freqüentemente construídos com o auxílio de uma matriz de substituição de pontos, a qual especifica uma pontuação para alinhamento de cada par de resíduos (aminoácidos).

Em virtude de proteínas remotamente (distantemente) relacionadas poderem compartilhar apenas regiões isoladas de similaridades, as medidas de similaridade de sequências locais passam a ter maior importância na descoberta de relacionamentos entre proteínas do que as medidas de sequências globais. A idéia básica é considerar apenas subsequências relativamente conservadas¹⁰, pois regiões dissimilares não irão contribuir ou subtrair as medidas de similaridade. A prática mais comum é considerar segmentos de todos os tamanhos, e escolher aqueles que otimizam as medidas de similaridade [63].

Para avaliar alinhamentos locais, *scores* (pontuações) geralmente são definidos para cada par de resíduos alinhados/coincidentes (o conjunto destes *scores*

¹⁰ Regiões locais (das sequências) similares na arrumação e na quantidade dos seus resíduos

é denominado matriz de substituição), como também para resíduos alinhados com intervalos (distâncias). A pontuação para o alinhamento global é resultante da soma destes pontos. A especificação uma matriz de substituição de aminoácidos adequada é primordial para os métodos de comparação de proteínas [19]. O principal objetivo é encontrar a matriz mais adequada a distinguir relacionamentos com distâncias evolutivas daqueles resultantes de similaridades casuais entre as sequências de proteínas, por isso novas definições, análises e refinamentos de matrizes vêm sendo realizados para desenvolver conjuntos de pontos otimizados para comparação local de sequências de proteínas [35].

3.1.4.1 Significância Estatística de Alinhamentos Locais de Sequências

Os alinhamentos globais somente são utilizados caso *gaps* sejam introduzidos, o que não é verdadeiro para alinhamentos locais. A habilidade de selecionar segmentos com posições iniciais arbitrárias (posições que podem ser manipuladas para inicializar um segmento) para cada sequênciasignifica que regiões biologicamente significantes podem, com frequência, ser alinhadas sem a necessidade de introduzir *gaps*. Os alinhamentos locais sem *gaps*, ou seja, duas sequências sendo comparadas com segmentos de igual tamanho recebem tratamento estatístico análogos (em termos de pontuações) aos alinhamentos que usam *gaps* [68].

Considera-se que os aminoácidos alinhados a_i e a_j recebem a pontuação de substituição S_{ij} . Duas sequências de proteínas com pares de segmentos de comprimentos equivalentes, quando alinhadas, têm a maior pontuação agregado denominado Par de Segmentos de Comprimento Máximo (PSM). Um PSM pode ser de qualquer comprimento; sua pontuação é a pontuação PSM.

Para se conhecer a maior pontuação PSM provavelmente obtida entre duas sequências relacionadas ou não relacionadas seria necessário algum modelo provável de ocorrência de aminoácidos numa sequência de proteína. O mais simples seria considerar que nas proteínas comparadas, o aminoácido a_i ocorre aleatoriamente com a probabilidade p_i . Estas probabilidades são escolhidas para refletir as frequências observadas dos aminoácidos nas proteínas atuais. Uma sequência de proteína aleatória seria construída de acordo com este modelo

3.1.4.2 Matrizes de Substituição Adequadas no Alinhamentos Locais de Sequências

Dada uma matriz de substituição qualquer e um modelo aleatório de proteína, efetua-se o cálculo do conjunto de frequências alvo, q_{ij} . Já, dentre os alinhamentos representantes de similaridades distantes (remotas), os aminoácidos são arrumados aos pares com determinadas frequências características.. Apenas se estas corresponderem às frequências alvo de uma matriz, utilizada na comparação, pode esta matriz ser considerada adequada para distinguir as similaridades locais distantes daquelas obtidas aleatoriamente (ou seja, daquelas que apesar de apresentarem similaridades, não são relacionadas/homólogas).

Variados modelos de matrizes de substituição de aminoácidos vêm sendo propostas para uso em programas de comparação de sequências. Muito embora, seja possível considerar que na busca de segmentos pares sem *gaps* e com pontuações altas, qualquer matriz que apresente uma distribuição da ocorrência/frequência de pares de aminoácidos que caracterize alinhamentos possa ser apontada como ideal para consultas. Ou mais precisamente, seja p_i a frequência na qual o aminoácido i ocorra em sequências de proteínas e, dentro do grupo dos alinhamentos encontrados, considere q_{ij} como a frequência alvo para que aminoácidos i e j sejam alinhados

numa determinada matriz. Então, as pontuações (*scores*) que melhor distinguem esses alinhamentos daqueles ocorrentes ao acaso (aleatórios) são dados pela fórmula.

$$S_{ij} = \left(\log \frac{q_{ij}}{p_i p_j} \right) \quad (1)$$

A base do logaritmo é arbitrária, afetando apenas a escala dos *pontos*, ou seja, a pontuação (o *score*) de um par de aminoácidos é o logaritmo (em determinada base) da frequência alvo desse par dividida pela frequência, na qual ele (o par) realmente ocorre no alinhamento. Essa razão compara a probabilidade de um evento ocorrer dentro das duas hipóteses de ocorrência. Qualquer conjunto de *scores* para alinhamento local pode ser definido dessa forma, assim a escolha de uma determinada matriz de substituição está diretamente condicionada a seleção de frequências alvo q_{ij} .

As matrizes são geradas a partir da observação de como e com que frequência os aminoácidos ocorrem (são arrumados) nas sequências de proteínas. Representam conjuntos ordenados utilizados para mensurar o grau de coincidência entre as sequências comparadas. Então quando é necessário comparar duas sequências, tais matrizes são utilizadas para atribuir pontos para as similaridades que podem ser encontradas entre as mesmas. Existem matrizes mais rigorosas do que outras, no tocante, a atribuição de pontos para as similaridades. Quanto mais rígida for a matriz, mais seletiva será a mesma nessa atribuição de pontos. Ou seja, para uma família de proteínas homólogas, seria necessária uma matriz de substituição que atribuisse pontos considerando apenas diferenças mínimas entre as proteínas comparadas. Enquanto que para proteínas relacionadas, mas de origens remotas, uma matriz menos rígida que considerasse alinhamentos curtos e distantes seria a mais indicada.

3.1.4.3 Pontuações de Alinhamentos Locais usadas como Medidas de Informação

Definido o número esperado de PSMs com score S e equivalente a p (a probabilidade de ocorrência/frequência de determinado aminoácido em proteínas), o score S poderia ser obtido através de:

$$S = \log_2 \frac{K}{p} + \log_2 N \quad (2)$$

Para matrizes de substituição típicas, valores de K entre 0-1, e de p entre 0-0,5 podem produzir alinhamentos significativos. A pontuação necessária para distinguir um PSM¹¹ (Par de Segmentos de Comprimento Máximo) daquele obtido aleatoriamente é aproximadamente igual ao número de bits necessários para especificar onde o PSM começa em cada um das duas sequências sendo comparadas. $\log_2 N$ bits de informação são usadas para distinguir entre N possibilidades.

Para a comparação de duas proteínas com tamanhos de 250 aminoácidos/resíduos cada, aproximadamente 16 bits de informação são necessários, para a comparação de uma sequência de proteína (250) com um banco de sequências contendo 4.000.000 de resíduos, em torno de 30 bits seriam necessários. Quando examinados dessa forma, as pontuações obtidas a partir dos alinhamentos não podem ser consideradas como simples valores arbitrários.

¹¹ Dadas duas sequências de proteínas, o par de segmentos com comprimentos equivalentes que, quando alinhado, obtêm o maior valor de pontos (*score*) denominado Par de Segmentos com Comprimento Máximo. Um PSM pode ser de qualquer comprimento, seu *score* é o *score* relativo ao PSM.

3.1.4.4 Seleção Adequada de uma Matriz de Substituição

De uma forma geral, o problema está em decidir que matrizes de substituição são as mais apropriadas para pesquisas em bancos de dados e para a comparação (detalhada) do pareamento de resíduos em sequências.

Dentre os PSMs obtidos pela comparação de sequências aleatórias, os aminoácidos a_i e a_j são alinhados com uma certa frequência de aproximação $q_{ij} = p_i p_j e^{\lambda S_{ij}}$. (p_i e p_j representam as probabilidades dos aminoácidos a_i e a_j aparecerem aleatoriamente em duas sequências de proteínas comparadas, já o S_{ij} é a pontuação (*score*) de substituição definida para o alinhamento destes aminoácidos) [35].

Dada uma matriz de substituição qualquer e um modelo aleatório de sequência de proteína, calcula-se o conjunto de frequências alvo características, q_{ij} , dos alinhamentos. Nos alinhamentos com similaridades remotas/distantes, os aminoácidos são pareados (arrumados aos pares devido a existência de similaridades) com certas frequências características. Apenas se estas corresponderem a uma matriz de frequências alvo, anteriormente definida, pode essa matriz ser considerada a mais adequada para distinguir similaridades locais distantes das similaridades daquelas obtidas por acaso. Uma informação importante a ser considerada seria o valor médio do *score* (bits de informação) obtido por par de resíduos nestes alinhamentos. Este valor pode ser definido como

$$H = \sum_{i,j} q_{ij} S_{ij} = \sum_{i,j} q_{ij} \log_2 \frac{q_{ij}}{p_i p_j} \quad (3)$$

onde H depende tanto da matriz de substituição quanto do modelo de sequenciade proteína escolhido. H é a entropia entre as frequências de distribuição alvo e as frequências de distribuição aleatórias.

3.1.5 Sensibilidade e Seletividade nas Comparações de Sequências

Em qualquer pesquisa a bases de dados, há sempre uma sequência que consegue uma melhor pontuação, independente se a sequência compartilha uma mesma sequência de origem ou devido a qualquer outra similaridade significativa com a sequência de consulta. Na comparação de sequências, existe uma estreita relação entre a **sensibilidade** (a habilidade de identificar sequências de origens distantes/remotas, mas correlacionadas) e a **seletividade** (a forma de evitar os falsos pontos positivos, ou seja, sequências não correlacionadas, mas com altos pontos de similaridade). O método perfeito para comparação de sequências deveria ser tanto sensitivo quanto seletivo; ele enfileiraria todos os aminoácidos de uma família de proteínas que compartilhassem uma sequência de origem comum em detrimento de quaisquer sequências que pudessem ser similares, mas não homólogas. Mas tais programas não existem, porque proteínas evoluem a taxas muito rápidas e diferenciadas, assim uma determinada sequência pode conter apenas um remoto traço de sua sequência ancestral. Na maioria dos casos, o problema se concentra em diferenciar as sequências com pontuações altas e que compartilham a mesma origem ou semelhanças significativas com a sequência de consulta, daquelas com pontuações relativamente altas também, mas sendo estas devido a composição local da sequência e a chances aleatórias. Existem programas mais sensitivos do que outros, ou seja, ideais para identificar sequências remotamente relacionadas. Contudo, um cuidado adicional deve ser tomado quando resultados de pesquisas desses programas forem analisados, porque métodos muito sensitivos são, em contrapartida, pouco seletivos.

3.1.6 Eficiência dos Métodos de Comparação

A eficiência desses métodos de consultas a bases de sequências moleculares é dependente de um grande número de fatores correlacionados, tais como:

Sistemas Verificadores de Acertos/Scoring - A grande maioria dos aplicativos de busca ordena os alinhamentos de sequências pela taxa de acertos (*scores*). Devido as características diversas (longas e diferentes / curtas e equivalentes) das sequências a serem pesquisadas, diferentes sistemas de pontuação, e adequados a cada situação, devem ser disponibilizados para os usuários.

Estatísticas sobre Alinhamentos/Associações - Dada uma sequência de consulta, a maioria dos programas de consulta a bases de dados produzem uma lista ordenada de semelhanças existentes entre a sequência de consulta e as sequências da biblioteca, alvo de análises estatísticas atualmente embutidas nos aplicativos de comparação.

Bases de Dados – Uma atualização constante das bases de sequências deve ser efetuada, visando evitar que novas descobertas sobre relacionamentos de sequências sejam perdidos pelo uso de bases fracas e desatualizadas. Deve-se evitar também a utilização de bases de dados com informações redundantes e sequências repetidas.

Embora as soluções biocomputacionais atuais sejam muito úteis na identificação de padrões e funções de proteínas e genes, eles ainda estão longe de serem perfeitos. Eles não só são demorados, exigindo superestações para rodarem, como também podem conduzir a falsas interpretações e suposições devido a simplificações necessárias. É então, ainda obrigatório, usar um raciocínio biológico e bom senso na avaliação dos resultados gerados por um programa de comparação de

sequências. Também, para avaliar o valor da saída de um programa é necessário compreender o fundo teórico matemático do mesmo para finalmente propor uma análise mais sensível.

3.2 Mineração de Textos

Mineração de textos, também chamado de mineração de dados textuais ou descoberta de conhecimento de bases de dados textuais é um campo novo e multidisciplinar que inclui conhecimentos de áreas como Informática, Estatística, Linguística e Ciência Cognitiva. Mineração de textos consiste em extrair regularidades, padrões ou tendências de grandes volumes de textos em linguagem natural, normalmente, para objetivos específicos. Inspirado no método data mining ou mineração de dados, que procura descobrir padrões emergentes de banco de dados estruturados, a mineração de textos pretende extrair conhecimentos úteis de dados não estruturados ou semi-estruturados.

O processo de Mineração de Textos também pode ser descrito como um processo de identificação de informações desconhecidas de uma coleção de textos [30]. Por informações desconhecidas, pode-se pensar em associações, hipóteses ou tendências que não estão explícitas no texto objeto de análise. Mooney e Nahm [49] descrevem mineração de textos como “uma procura por padrões em textos não-estruturados”. Já Dorre et al. [20] afirmam que “mineração de textos aplica as mesmas funções de análise de mineração de dados para o domínio de informações textuais, baseado em sofisticadas técnicas de análise de textos que obtém informações de documentos não-estruturados”, enquanto Tan [64] descreve mineração de textos como “o processo de extração de padrões interessantes e não triviais ou conhecimento de documentos textuais”. Nenhum dos autores, todavia, enfatiza explicitamente a importância das novidades descobertas nas informações mineradas.

A extração de informações está diretamente envolvida no processo de mineração de textos. Por exemplo, uma abordagem utilizada na mineração de textos

semi-estruturados na *web*, é usar técnicas de extração para converter documentos em uma coleção de dados estruturados e depois aplicar técnicas de mineração de dados para análise.

A mineração de textos representa, de fato, um avanço em relação à recuperação de textos. É uma área de pesquisa relativamente nova que está mudando a ênfase das tecnologias de informações baseadas em texto, saindo de um nível de recuperação e extração, para um nível mais alto, como análise e exploração dos dados. Considerando a grande quantidade de informações disponíveis em forma de texto atualmente, ferramentas que realizem atividades nesta área de forma automática, ou que simplesmente auxiliem os usuários a fazê-las, são extremamente bem-vindas.

A categorização de textos, que em sua forma mais simples, consiste na classificação binária (*single-label*), onde é possível classificar os documentos em apenas uma de duas categorias.

Em classificação binária, a classe de interesse é chamada de classe positiva e os documentos desta classe são denominados documentos relevantes ou exemplos positivos. Já as demais classes representam a classe negativa e, os seus documentos são denominados como irrelevantes ou exemplos negativos.

De forma geral um processo de mineração de textos contém quatro macro etapas: coleta, pré-processamento, indexação e análise da informação. Nos próximos parágrafos é descrito o processo como um todo e em seguida cada uma das etapas de forma mais detalhada.

A etapa inicial tem por objetivo a coleta de das informações que vão compor a base textual de trabalho, isto é, determinar e selecionar o universo de atuação das

técnicas de mineração de texto. Por outro lado, nenhuma informação que não esteja contida na base textual poderá ser extraída, encontrada ou utilizada de alguma forma.

Após a coleta de documentos é necessário transformar os documentos em um formato propício para serem submetidos aos algoritmos de extração automática de conhecimento. Essa segunda etapa, denominada de pré-processamento, é responsável por obter uma representação estruturada dos documentos, geralmente no formato de uma tabela atributo-valor.

Essa tabela atributo-valor que representa os documentos tem como característica valores esparsos dos dados e uma alta dimensionalidade. Essas características são inerentes a problemas relacionados ao processo de MT, pois cada palavra presente nos documentos pode ser um possível elemento do conjunto de atributos dessa tabela atributo-valor. É, portanto, uma etapa bastante custosa e um cuidadoso pré-processamento dos documentos são imprescindíveis ao sucesso de todo o processo de MT.

Após os documentos serem representados em um formato adequado, é possível aplicar técnicas de extração de conhecimento utilizando sistemas de mineração de dados. Caso os documentos estejam representados no formato de uma tabela atributo-valor, geralmente, na terceira etapa, são empregados métodos de RI como indexação para aumentar a *performance* do processo.

Finalmente, na última etapa, o objetivo é descobrir padrões úteis e desconhecidos presentes nos documentos. Para a extração de padrões, são utilizadas técnicas de forma semelhante ao processo tradicional de MD.

3.2.1 Análise dos Dados

Antes do início das atividades de mineração de textos é importante que se saiba como as informações são tratadas em um texto. Duas formas de abordagem podem ser utilizadas: a análise semântica que é baseada na funcionalidade dos termos nos textos e a análise estatística que é baseada na frequência em que os termos aparecem nos textos (Ebecken et al.[22]).

Dentre estas abordagens, destacamos as análises: semântica e estatística. Na análise semântica, as informações textuais são trabalhadas conforme a sequência em que os termos aparecem no contexto da frase, identificando a correta função de cada termo. Com isso é possível identificar a real importância de cada termo em determinados contextos, possibilitando um ganho na qualidade dos resultados produzidos.

Na Análise Semântica utilizam-se fundamentos e técnicas baseadas no processamento de linguagem natural. Para a compreensão da linguagem natural é importante que se entenda os principais tipos de conhecimentos existentes. Dentre eles estão: o conhecimento morfológico, sintático, semântico e pragmático [10].

Na Análise Estatística a importância dos termos está diretamente ligada à quantidade de vezes em que eles aparecem nos textos. Este tipo de análise permite, independente do idioma, a criação de modelos através do aprendizado estatístico, capazes de fornecer soluções computacionais suficientemente precisas para tarefas em que a aplicação de mão-de-obra humana é inviável devido à complexidade da tarefa ou do grande volume de dados (Ebecken et al. [22]).

A seguir, então, explicaremos de forma mais detalhada cada uma das etapas envolvidas no processo de mineração de texto.

3.2.2 Coleta de Dados

A coleta de dados tem como função formar a base textual de trabalho. Essa base pode ser estática, nos casos mais simples, ou dinâmica, isto é, atualizadas a todo momento através de robôs autônomos coletando novas informações. A atualização é feita pela simples adição de um novo conteúdo, remoção de conteúdos antigos, ou, substituição da base por uma inteiramente nova.

Coletar dados é uma atividade trabalhosa. Um dos motivos é que os dados podem não estar disponíveis em um formato apropriado para serem utilizados no processo de mineração de textos.

Para mineração de textos, um dos principais problemas em coletar dados é descobrir onde os dados estão armazenados. Depois disto, recuperar documentos relevantes ao domínio de conhecimento. De forma geral, esse procedimento se estabelece basicamente em três ambientes distintos: no diretório de pastas do disco rígido; em tabelas de diferentes bancos de dados e na Internet.

Trabalhos relacionados à coleta de documentos provenientes da Internet podem ser encontrados na literatura [7]. Muitos deles combinam técnicas de AM e Recuperação de Informação (RI) [56] para determinar o perfil do usuário visando melhorar a coleta de documentos.

3.2.3 Pré-processamento

O pré-processamento de textos consiste em um conjunto de transformações realizadas sobre alguma coleção de textos com o objetivo de fazer com que esses passem a ser estruturados em uma representação atributo-valor. De modo geral, a etapa de pré-processamento tem por finalidade melhorar a qualidade dos dados já disponíveis e organizá-los. As ações realizadas na etapa de pré-processamento de dados visam prepará-los para serem submetidos a algum algoritmo de indexação ou mineração de dados.

Conceitualmente, em um processo de mineração, essas transformações consistem em identificar, compactar e tratar dados corrompidos, atributos irrelevantes e valores desconhecidos [72].

Em mineração de textos, pré-processamento normalmente significa dividir o texto em palavras, realizar uma análise léxica (gerar um thesaurus), remover as *stopwords* (termos considerados irrelevantes) aplicar técnicas de *stemming* (normalização morfológica dos termos), e classificá-las segundo a classe gramatical.

No entanto, a etapa de pré-processamento vai além das ações citadas, pois é necessário transformar os textos em uma representação estruturada adequada para que, a partir disso, os dados possam ser submetidos ao processo como um todo. Durante a transformação dos textos em formato estruturado existe a possibilidade de que informação intrínseca ao conteúdo dos textos seja perdida. Um desafio, nesse caso, é obter uma boa representação minimizando a perda de informação. A etapa de pré-processamento em um processo de MT é, portanto, fundamental para o desempenho de todo o processo [28].

Thesaurus: É uma lista de palavras com significados semelhantes, a partir de um domínio específico, o que o torna restrito, pois seu objetivo é apresentar as diferenças mínimas entre as palavras, permitindo assim que aplicar substituir os termos que tem relação no thesaurus. O thesaurus não inclui as definições dos termos.

O *thesaurus* pode ser definido como um vocabulário controlado que representa sinônimos, hierarquias e relacionamentos associativos entre termos para ajudar os usuários a encontrar a informação que eles precisam [22].

Stopwords: Um dos pontos importantes da preparação dos dados é a eliminação das palavras que não possuem representatividade sobre o texto. Essas palavras normalmente possuem alta incidência em documentos distintos e não devem ser usadas para representar diferenças ou similaridades entre documentos, uma vez que são muito comuns. O conteúdo semântico dessas palavras é insignificante e não são relevantes na análise dos documentos. Ao se retirar estas palavras do índice, reduzimos o espaço de armazenamento e melhoramos seu desempenho durante a tarefa de *Text Mining*. Uma lista básica de palavras pode ser usada para formar uma lista de *stopwords* chamada de *stoplist*, de forma a melhorar a tarefa de *Text Mining*.

A maioria dessas palavras são conjunções, preposições ou pronomes que pouco contribuem para a formação do índice [14].

Stemming: O processo de *Stemming* em *Text Mining* visa reduzir determinados tipos de palavras ao seu radical (raiz), para que diferentes palavras com o mesmo significado não sejam contadas como palavras distintas. Normalmente, é usada uma palavra em uma consulta e as respostas podem trazer documentos não relevantes ao contexto exigido. Acontece que uma variante dessa palavra pode ser encontrada em

outros documentos com maior relevância. Sendo assim, a consulta pode ser mais eficiente se trabalhar com os radicais das palavras ao invés da palavra original. Uma mesma palavra pode ter variações como plurais, formas de gerúndio e acréscimo de sufixos, por exemplo [18]. A técnica consiste então em remover os sufixos e prefixos das palavras para que palavras com o mesmo radical tenham significados similares [24].

Os algoritmos de *stemming* mais conhecidos são os algoritmos de Porter [55], Lovins [45] e Paice [52] que removem sufixos da língua inglesa. Geralmente utiliza-se o radicalizador de Porter [55], desenvolvido especificamente para a língua inglesa. Como tal técnica não se adapta bem à língua portuguesa, optou-se pela implementação do algoritmo *Portuguese Stemmer*, proposto por Orengo e Huyck [51]. Apesar destes algoritmos serem utilizados com o objetivo de otimizar o desempenho dos processos de recuperação da informação (RI) e mineração de textos, alguns estudos demonstram que a redução de afixos em uma coleção de documentos não apresenta uma melhora significativa no desempenho da tarefa de mineração aplicada [29] [40], porém somente uma redução no espaço de representação dos dados.

3.2.3.1 Geração de Vetores para Predição de Dados

As características dos documentos são as palavras ou os *tokens* que eles contêm. Assim, somente a partir de análise profunda do conteúdo linguístico dos documentos, pode-se descrever cada documento que representam os *tokens* mais freqüentes.

Esta coleção de características/palavras é normalmente denominada dicionário. Os tokens/palavras do dicionário formam a base para criação de uma planilha de dados numéricos correspondentes a coleção de documentos. Cada linha é um documento, e

cada coluna representa uma característica. Assim, uma célula na planilha é uma medida de uma característica (correspondente a coluna) para um documento (correspondente a uma linha). Os métodos preditivos aprendem com estes dados. No modelo mais básico de tais dados, ocorre uma simples verificação da presença e da falta dos termos/palavras, e as entradas nas células são entradas binárias correspondentes a um documento e a uma palavra. O dicionário de palavras cobre todas as possibilidades e corresponde ao número de colunas da planilha. As células receberão 1's ou 0's, dependendo se as palavras são encontradas ou não no documento. A Figura 4-a ilustra um exemplo de uma planilha de características de documentos.

Se um método de aprendizagem pode lidar com dicionários de grandes dimensões, este simples modelo de dados precisa ser muito eficiente. A verificação das palavras é simples porque não é realmente checada cada palavra do dicionário. Uma tabela *hash* (*hash table*) do dicionário é contruída e, então, é verificada a existência das palavras do documento nesta tabela. Este método garante que muitas variações e combinações de palavras serão recuperadas com maior agilidade em grandes amostras de documentos.

Todavia, em outras circunstâncias, é preferível trabalhar com dicionários menores. A amostra pode ser relativamente pequena, ou um grande dicionário pode ser muito pesado. Em tais casos, pode-se tentar reduzir o tamanho do dicionário por meio de inúmeras transformações de um dicionário e de suas palavras constituintes. Dependendo do método de aprendizagem, muitas destas transformações podem aprimorar o desempenho da predição.

Se a predição for o principal objetivo, uma outra coluna é necessária para indicar a resposta (ou classe) correta para cada documento. Na preparação dos dados para um método de aprendizagem, esta informação estará disponível nos rótulos dos documentos. Os rótulos são geralmente binários, e a menor classe é quase sempre a de maior interesse. Ou seja, ao invés de gerar-se um dicionário global para ambas as classes, consideram-se apenas as palavras encontradas na classe, a qual se tenta prever. Se esta classe é bem menor que a classe negativa, o que é típico, um dicionário local será bem menor do que um dicionário global.

Outras formas de redução de um dicionário são remover *stopwords* e palavras frequentes, utilizar *stemming* e seleção de características. Esta última define a utilização de estratégias que tentam selecionar, a partir do dicionário global de uma coleção de documentos, um subconjunto de palavras que parecem ter maior potencial de predição. Estes métodos de seleção são frequentemente complicados e independentes do método de predição. Geralmente, opta-se pela utilização da informação da frequência das palavras, a qual é mais fácil de determinar-se. Muitos métodos de predição de texto têm sido ajustados para lidar, preferivelmente, com grandes dicionários do que com a geração de vários dicionários menores. Se muitas classes devem ser determinadas, então a geração de um dicionário menor deve ser repetida para cada problema de predição. Por exemplo, se existem 100 tópicos para serem categorizados, então há 100 problemas de predição binária para serem resolvidos. As escolhas são 100 dicionários pequenos ou um único grande. Normalmente, os vetores de uma planilha modelo serão novamente gerados para corresponder ao pequeno dicionário.

Embora, os dados sejam descritos a partir do preenchimento de uma planilha, espera-se que a maioria das células contenha zeros. Uma grande parte dos documentos contém um pequeno subconjunto das palavras do dicionário. No caso da classificação de textos, uma coleção de textos pode conter milhares de tipos de palavras. Cada documento individual pode conter poucas centenas de *tokens* únicos. Assim, na planilha, quase todas as entradas para aquele documento serão zeros. Preferível a armazenar todos estes zeros, a melhor alternativa é representar a planilha como um conjunto de vetores esparsos, onde uma linha é representada por uma lista de pares, um elemento do par sendo um número de coluna e o outro sendo o valor *nonzero* da característica correspondente. A Figura 4-b apresenta um exemplo simples de com uma planilha pode ser transformada em vetores esparsos. Todas as representações de dados propostas devem estar consistentes com as representações de dados esparsos.

Company	Income	Job	Overseas
0	1	0	1
1	0	1	1
1	1	1	0
0	0	0	1

Figura 4-a - Uma planilha binária de palavras nos Documentos

Planilha				Vetores Esparsos	
0	15	0	3	(2,15)	(4,3)
12	0	0	0	(1,12)	
8	0	5	2	(1,8)	(3,5) (4,2)

Figura 4-b - Planilha para vetores esparsos

3.2.4 Redução da Dimensionalidade

Um dos maiores problemas de mineração de texto é lidar com espaços de dimensão muito alta se considerado um espaço-vetorial onde cada termo representa

uma dimensão, teremos tantas dimensões quanto palavras diferentes. Dessa forma, um dos problemas importantes tratados no pré-processamento dos dados é reduzir o número de termos.

Uma estratégia bastante citada na literatura é a utilização da Lei de Zipf [82] e cortes de Luhn [46]. A técnica tem caráter estatístico e funciona da seguinte forma:

A Lei de Zipf diz que se f é a frequência de ocorrência de qualquer palavra do texto, e r a posição de ordenação com relação às outras palavras então produto $f \times r$ é aproximadamente constante. Luhn propôs que, em um gráfico f versus r pode-se definir um limite superior e um limite inferior de corte. As palavras que estiverem fora do intervalo são excluídas da análise. Na Figura 5 é mostrada uma ilustração desse procedimento.

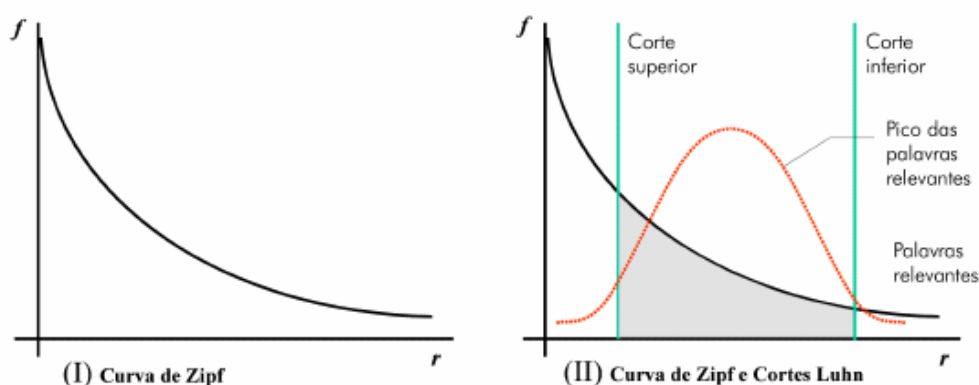


Figura 5 – A curva de Zipf e os cortes de Luhn

3.2.5 Preparação dos Dados

Esta etapa envolve a identificação e seleção, nos dados pré-processados, dos termos que melhor expressam o conteúdo de textos, ou seja, toda e qualquer informação que não refletir nenhuma idéia considerada importante deve ser desconsiderada.

Desta maneira, esta seleção reduz a quantidade de termos e, por conseguinte, a dimensão dos vetores que representam os documentos. A redução da dimensionalidade implica na redução da utilização de memória, bem como na redução da carga de processamento. Além disso, reduz a possibilidade de “*overfitting*”, fenômeno que ocorre quando o classificador é ajustado de maneira muito específica para o conjunto de treinamento, implicando em uma baixa *performance* na classificação de documentos não conhecidos pelo classificador.

A preparação dos dados é a primeira fase do processo de criação indutiva de classificadores de texto. Deve-se entender por indução a possibilidade de obtenção de conclusões genéricas sobre um conjunto particular de exemplos (conjunto de treinamento). O raciocínio indutivo é utilizado para generalizar conceitos, aproximar funções e descobrir novas funções através de exemplos fornecidos.

Com o objetivo de reduzir a quantidade de termos, algumas etapas são realizadas, visando um melhor desempenho do sistema na manipulação dos documentos.

De acordo com Meadow et al. [47], muitos termos não possuem significado semântico suficiente para descrever o assunto de um texto. Desta maneira, é interessante que se use uma medida para calcular a relevância dos termos nos documentos.

Os métodos mais utilizados para o cálculo de relevância são os baseados na frequência de termos [56] [58], são eles:

- (i) Frequência absoluta - mais conhecida como *term frequency*, indica o número de ocorrências de uma palavra em um documento. É normalmente denominada de *tf*.
- (ii) Frequência relativa - consiste na frequência absoluta (*tf*) normalizada pelo tamanho do documento (número de palavras nele contidas). A frequência relativa (*normalized term frequency – ntf*) de um termo *x* em um documento qualquer, pode ser calculada dividindo-se a frequência do termo *tf* pelo número total de termos no documento *N*, ou seja, $Frelx = tf(x)/N$
- (iii) Frequência inversa de documentos - Uma das maneiras mais utilizadas para se identificar o peso do termo é a aplicação da fórmula *tf - idf* = $Freq_{td}/DocFreq_t$, sendo que $Freq_{td}$ é o número de vezes em que o termo *t* aparece no documento *d* e $DocFreq_t$ corresponde ao número de documentos em que o termo *t* aparece.

Com esse cálculo é possível indicar a quantidade de documentos em que um termo ocorre, baseando-se no cálculo da frequência do termo e na remoção do espaço de características dos termos. Ou seja, a partir da frequência dos termos e da frequência dos documentos é possível obter a *frequência do termo - frequência inversa de documentos*, mais conhecida como *tf-idf*. Através desta medida pode-se aumentar a importância de termos que aparecem em poucos documentos e diminuir a importância de termos que aparecem em muitos documentos.

Baseado na relevância das palavras nos textos, é possível realizar a codificação dos dados e escolher um modelo de representação de documentos [59]. A codificação mais frequentemente utilizada é a sugerida na técnica *bag of words* [18], na qual cada documento é representado por um vetor composto de termos presentes no documento. Esta codificação pode ser considerada uma simplificação das informações expressas por um documento, não fornecendo, portanto, uma descrição fiel do conteúdo.

Alguns modelos de representação de documentos foram desenvolvidos na área de Recuperação de Informação (RI). Os mais utilizados são o modelo Booleano e o modelo de Espaço de Vetores, propostos por Baeza-Yates e Ribeiro-Neto [7].

3.2.6 Similaridade entre termos

Um sistema de Recuperação de Informações tem como base a seguinte teoria [59] perguntas (consulta) são submetidas pelo usuário, perguntas estas baseadas em termos (palavras) que identificam a idéia desejada por este usuário; os documentos são identificados pelos termos que eles contêm, portanto, a localização de um documento desejado pelo usuário dá-se a partir da identificação da similaridade entre o(s) termo(s) fornecido(s) pelo usuário e os termos que identificam os documentos contidos na base de dados. A Figura 6 representa esquematicamente esta teoria:



Figura 6 - Função Similaridade

No processo de recuperação da informação a similaridade entre termos é fundamental. A determinação da similaridade ocorre entre documentos e consultas submetidas à base de documentos, e a recuperação pode ser exata ou aproximada. Entre os modelos de comparação destacam-se: [41]

- 1) Modelo Booleano:** No modelo booleano os documentos recuperados são aqueles que contêm os termos que satisfazem a expressão lógica da consulta. Uma consulta é considerada como uma expressão booleana convencional formada com os conectivos lógicos *AND*, *OR* e *NOT*. Pelo fato do modelo ser binário, a frequência de um termo não tem efeito. Uma grande vantagem deste modelo é a sua simplicidade e a necessidade de pouco espaço de armazenamento.

Os problemas relacionados ao modelo booleano são, no entanto, bem conhecidos. A formulação de uma consulta adequada é difícil, isto é, a seleção dos termos para a consulta é difícil. Além disso, sem um grau de comparação parcial, não se pode saber o que foi deixado de fora da definição da consulta. Uma vez que não há um grau de comparação, não é possível ordenar os resultados de acordo com a relevância.

2) Modelo Vetorial: representa documentos e consultas como vetores de termos. Termos são ocorrências únicas nos documentos. A organização e representação dos documentos textuais neste modelo são feitas através de vetores de termos-índices que descrevem as propriedades de cada documento em particular. Os documentos devolvidos como resultado para uma consulta são representados similarmente, ou seja, o vetor resultado para uma consulta é montado através de um cálculo de similaridade.

O modelo de espaço de vetores, conhecido com SVM, visa contornar as limitações do modelo booleano, com a utilização de pesos e, desta maneira, permitir uma similaridade parcial entre os documentos e as palavras.

No modelo de espaço de vetores, como na maioria dos modelos de recuperação de informação, a expressão de igualdade entre termos é representada através do que é chamado de similaridade. Então, adota-se este termo como forma de representação do número de propriedades (ou características) que determinados objetos (documentos ou termos) possuem em comum, assim como o número de propriedades incomuns entre objetos.

3) Modelo Probabilístico: descreve documentos considerando pesos binários que representam a presença ou ausência de termos. O vetor resultado gerado

pelo modelo tem como base o cálculo da probabilidade de que um documento seja relevante para uma consulta. A principal ferramenta matemática do modelo probabilístico é o teorema de Bayes [51].

3.2.7 Indexação

As técnicas de indexação de documentos foram bastante difundidas pela demanda e crescimento da área de Recuperação de Informações (RI). Muitas pessoas acreditam que a área de recuperação automática de informações textuais é uma área nova. Esta idéia talvez tenha surgido com a Web (um dos serviços oferecidos pela Internet), onde milhares de informações, dispostas em forma de páginas (documentos textuais), estão disponíveis. No entanto, segundo [38] há aproximadamente 4000 anos já são praticadas técnicas de catalogação manual por índices.

Dentre os recentes trabalhos, os algoritmos de mineração de textos também reutilizam técnicas eficientes de indexação para manipulação eficiente dos textos. Como as técnicas de indexação permitem uma busca rápida por palavra-chave em grandes volumes de textos, existe ganho de *performance* que viabiliza cálculos estatísticos mais sofisticados e, por isso, potencialmente melhores.

No entanto, mineração de textos é um conceito bem mais extenso do que busca eficiente por palavras-chave. Por exemplo, uma busca por palavra-chave na Internet retornaria uma lista de páginas que contém os termos procurados, desconsiderando aspectos semânticos que podem tornar estas ocorrências irrelevantes para o objetivo proposto.

Técnicas de mineração de textos promovem análises mais extensas do conteúdo dos documentos, identificando fatos, relações e padrões de forma a obter uma percepção

similar àquela tida por um humano lendo o mesmo documento. A informação extraída é geralmente mais relevante, e pode ser usada para diferentes propósitos como categorizar um documento, identificar o significado ou o grupo semântico de expressões dentro do documento, auxilia a leitura de grandes volumes de texto.

Em um banco de dados textual, os dados não estão distribuídos de forma tabular. Até mesmo porque o texto é uma sequência de caracteres, não existindo uma pré-especificação de atributos. Não há como saber o que é um nome em um documento, a não ser que se faça uma análise de Linguagem Natural e se descubra o que pode vir a ser um nome. Logo, para localizar as informações sobre determinada pessoa em um banco de dados textual, seria necessário analisar caractere-por-caractere do texto até que a sequência de caracteres correspondente ao nome fosse localizada.

3.2.7.1 Busca caractere-a caractere

Uma palavra pode ser entendida como uma cadeia de caracteres específica que se deseja procurar. A informática se apropriou de muitos termos em inglês no seu vocabulário para representar tecnologias específicas, neste caso o termo importado e mais utilizado é *string*. Cadeias de caracteres implementadas no computador podem ser tratadas como *strings*. Sendo assim, procurar por uma *string* no corpo de um texto é preocupação fundamental para qualquer aplicação de mineração de textos.

Inúmeros trabalhos para resolver este problema foram propostos, e eles são comparados normalmente pela complexidade computacional do pior caso e o número de comparações de caracteres feitos. Para o problema de procurar todas as ocorrências de uma dada string de m caracteres em um texto de n caracteres, o pior caso é expresso pelo número de operações $c(n,m)$. Um primeiro bom resultado estabelecido foi dado por Knuth-Morris-Pratt (KMP) [37] como $c(n,m) = 2n-m+1$. Na mesma época Boyer-Moore (BM) [9] desenvolvia um algoritmo que alcançava apenas $c(n,m)=6n$, porém, ele foi

sendo melhorado nos anos seguintes chegando a $c(n,m) = (3n-n/m)$ por Cole [12]. No entanto uma variante de Boyer-Moore foi desenhada por Apostolic [5] igualando o KMP com $c(n,m) = 2n-m+1$. Em 1990, Colussi, Galil e Giancarlo [13] criaram um híbrido de KMP e BM para atingir a marca de $7n/6 \leq c(n,m) \leq (4n-m)/3$. Hoje um algoritmo bastante utilizado é o Karp-Rabin [36] que utiliza a função *hash*. A função *hash* codifica as cadeias (*strings*) em números e com isso ganha uma vantagem significativa por utilizar cálculos numéricos.

3.2.8 Predição

As predições de palavras definem imagens de decisões momentâneas e processos complexos cheios de imprecisões. Partindo de um perspectiva estatística, é um problema simples que tem solução. Entretanto, a solução nem sempre é a melhor. O problema está ilustrado na Figura 7. Baseado em uma amostra de exemplos de experiências anteriores, pode-se projetar novos exemplos. Se o futuro é similar ao passado, é possível realizar predições com precisão. Um exemplo de tal situação é quando se tenta prever o futuro preço das ações de um empresa baseado em registros históricos e em outras medidas de desempenho.

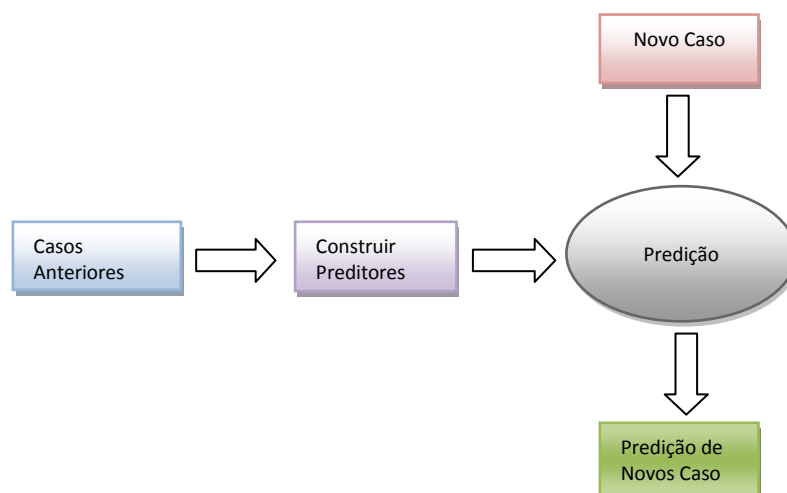


Figura 7 – Predição do futuro baseado no passado

Fazer previsões requer muito mais do que uma pesquisa a experiências anteriores. Mesmo que se consiga caracterizar estas experiências de uma forma consistente, o teste do sucesso é baseado em novos exemplos. Para a predição, um padrão deve ser encontrado em experiências passadas e se perpetuar no futuro, levando a resultados precisos sobre novos exemplos. Se um novo exemplo apresentar-se de uma forma radicalmente diferente de experiências anteriores, a própria aprendizagem por experiências passadas provará ser inadequada. Aprendizado de máquina e métodos estatísticos não aprendem baseados em princípios básicos. Eles não têm capacidade de raciocinar e chegar a novas conclusões sobre novas situações. Se as amostras podem ser obtidas e organizadas em um formato correto, encontrar padrões se torna quase fácil, mesmo para vetores de características multidimensionais.

O problema clássico de predição para textos é a categorização de textos. Um conjunto de categorias é predefinida, e o objetivo é associar uma categoria/tópico a um novo documento. Por exemplo, coleta-se vários artigos e define-se um conjunto de categorias/tópicos: financeiro, esporte, etc. Um conjunto de categorias é estabelecido. Quando novos artigos chegarem, suas palavras vão ser examinadas e tais artigos serão associados às possíveis categorias do conjunto predefinido.

O problema da predição de textos é geralmente um problema de classificação. Em um modelo de planilha de dados, geralmente, uma linha é um exemplo e uma coluna é um atributo a ser medido. Para a classificação, uma coluna adicional identifica a resposta correta. Tal resposta indica se o texto exemplo para determinada categoria é verdadeiro ou falso.

A planilha da 8-a representa a visão idealizada das palavras que formam um determinado padrão. Para predição, examinam-se padrões relativos a um rótulo, onde o rótulo é a resposta correta. O rótulo é verdadeiro ou falso, respectivamente

representados por 0 ou 1 na última coluna. Observa-se que as mesmas duas palavras (Word2 e Word4) sempre ocorrem para a classe 1 e nunca ocorrem juntas para a classe 0. Um padrão é formado quando uma combinação de palavras ocorre para a classe de interesse e não ocorrem para a classe negativa.

Para quaisquer aplicações de mineração de dados, a precisão das predições é dependente da qualidade preditiva dos atributos. Para a mineração de textos, a qualidade depende das palavras. Nem todos os rótulos podem ser discriminados. Em contraste com a Figura 8-a, na Figura 8-b, não existem preditores óbvios para a classe 1. Nenhum padrão é encontrado para separar as duas classes. Para um padrão ideal usando apenas uma única palavra, um valor 1 é encontrado na célula da palavra quando a classe for 1, e um valor 0 quando a classe for 0.

Word1	Word2	Word3	Word4	Word5		WordN	Label
0	1	1	1	0		1	1
1	0	1	0	0		1	0
1	1	0	1	1		0	1
0	1	0	0	1		1	0
1	1	1	1	0		1	1
0	1	0	1	1		0	1
1	0	1	1	0		1	0
0	1	1	1	1		0	1

Figura 8-a– Padrões preditivos em uma planilha

Word1	Word2	Word3	Word4	Word5		WordN	Label
0	1	0	1	0		1	1
1	0	1	0	0		0	0
1	0	0	1	1		0	1
0	1	0	0	1		1	0
1	0	0	1	0		1	1
0	1	0	0	1		0	1
0	0	1	1	0		0	0
0	0	1	0	1		0	1

Figura 8-b – Planilha sem padrões óbvios

3.2.8.1 Classificação de Documentos

A classificação de documentos é baseada no aprendizado e na realimentação a partir de uma avaliação dos resultados. Uma amostra é coletada. Os dados são

organizados em um formato estruturado. Qualquer exemplo é medido da mesma maneira. A resposta é expressa em termos de verdadeiro ou falso, a decisão é binária. Em termos matemáticos, a solução é uma função que mapeia exemplos de rótulos, $f: w \rightarrow L$, onde w é um vetor de atributos e L é um rótulo. Neste caso, os atributos são palavras ou *tokens*. Os rótulos podem ser o objetivo, o qual está potencialmente relacionado às palavras. A classificação ocorre a partir do método de aprendizado que aprende a partir de uma planilha modelo que pode prever os valores da última coluna (coluna adicional) para uma segunda planilha. O método necessita encontrar uma generalização ou padrão na planilha modelo que será utilizado para classificar os exemplos da segunda planilha.

3.2.8.2 Similaridade entre Documentos

Entre textos, uma medida muito usual é a similaridade. Ou seja, à primeira vista, um documento pode ser considerado similar ao outro se compartilham as mesmas palavras. A medida mais elementar de similaridade é contabilizar o número de palavras que dois documentos têm em comum. Pode-se então classificar os documentos por similaridade, onde os documentos mais similares são aqueles que compartilham a maioria das palavras. Observe que apesar das métricas tradicionais para cálculo da distância, este ignora palavras que ocorram em um documento mas que não ocorram no outro. O único interesse está nas palavras que ocorram em ambos os textos.

Usando uma planilha, quando um novo documento é apresentado, ele é comparado a qualquer documento já armazenado. Para qualquer palavra positiva no novo documento, contabiliza-se seu número de ocorrências nos documentos armazenados. A similaridade, S , é o número de palavras positivas encontradas em ambos, um documento armazenado e o novo documento. O uso de apenas palavras

positivas para a similaridade quase elimina a influência dos zeros, justificando o uso de representações esparsas.

3.2.9 Extração do Conhecimento

A etapa de extração do conhecimento é direcionada ao cumprimento dos objetivos definidos na identificação do problema, a partir da execução de um ou mais algoritmos.

Tipicamente, a extração de informação envolve a identificação de padrões que representam um contexto chave dentro do texto. Além disso, a EI utiliza um conjunto de filtros que, junto com os padrões, representam, de forma estruturada, a informação contida em cada texto, possibilitando a atualização de uma base de dados ou a melhora de uma recuperação de informações posterior [72].

Dentre as diversas técnicas implementadas, pode-se destacar:

Sumarização: É o processo da redução da quantidade de texto em um documento, porém mantendo seus significados-chave [43]. O sumário de um documento é derivado de um texto fonte condensado pela seleção e/ou generalização em palavras ou frases chaves presentes em textos o qual mantém o conteúdo informativo do documento original.

Categorização: Visa identificar os tópicos principais em documento e associar este documento a uma ou mais categorias/classes pré-definidas [50]. A categorização de documentos é uma tarefa bastante utilizada no caso de existir um conjunto de documentos previamente classificados.

Agrupamento (Clustering): É uma técnica bastante útil quando se quer agrupar documentos similares. O processo de *clustering* visa obter uma divisão útil de n objetos

em um número de clusters. Um *cluster* é uma coleção de objetos de dados que são similares uns aos outros, baseados em seus valores de atributos, e assim podem ser tratados coletivamente como um grupo. As técnicas de agrupamento de textos levam em conta as palavras que aparecem nos documentos para definir a função de similaridade a fim de determinar o agrupamento final [53].

Na literatura encontramos várias técnicas de clustering de documentos, tais como *Clustering Hierárquico*, *K-means* e *SOM*.

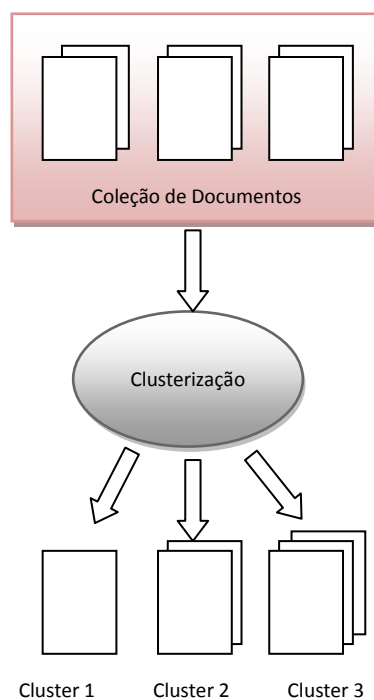


Figura 9– Organização de documentos em grupos

3.2.10 Categorização

Categorizar é classificar automaticamente documentos em relação a um conjunto de categorias ou matérias previamente conhecidas. Classificar é colocar um documento numa classe, dentro de um esquema, ou agrupar os documentos com características comuns, em função dos seus conteúdos.

Para criar categorias é necessário que cada uma delas possa ser representada por um termo ou conjunto de termos que suportam o significado do conceito associado. Metodologicamente, o problema reside na identificação de um termo ou conjunto de termos cujo significado semântico possa definir o conceito de uma determinada categoria. A indexação tem duas etapas fundamentais: a análise e a representação. Através da análise identificam-se os conceitos que constituem unidades de pensamento do conteúdo temático de um documento, expressas em linguagem natural. No entanto, conceito e termo não se identificam, na medida em que um conceito é uma idéia ou noção, a representação intelectual de um objeto, enquanto que o termo é a sua representação formal e o seu suporte visível. Por isso é necessário estabelecer, com rigor, as regras de associação dos termos aos conceitos. Para essa finalidade foram criadas as linguagens controladas de indexação. Uma vez selecionado ou criado o instrumento de linguagem controlada, os termos tidos como representativos do conteúdo do texto são expressos em linguagem de indexação, ficando resolvidas algumas das ambigüidades da linguagem natural. Esta representação pode ser vocabular – simples ou composta – ou simbólica. Finalmente os termos de indexação são integrados na respectiva categoria. O termo de indexação tem como característica principal a sua funcionalidade: atua na pesquisa como ponto de acesso à informação, como uma porta de entrada, por assunto, num sistema informativo. Este sistema prevê a interação do usuário para classificar a categoria determinada por um conjunto de termos.

Em resumo, a categorização de textos é uma ferramenta utilizada para classificar automaticamente um conjunto de documentos numa ou mais categorias preexistentes, não tendo outra finalidade senão recuperar a informação. Esta técnica é usualmente utilizada, num sentido mais amplo, para filtrar e encaminhar mensagens de correio eletrônico, organizar notícias, resumos e arquivos de publicações periódicas.

3.2.10.1 Classificador Bayesiano

O classificador *Naive Bayes* (NB) é provavelmente o mais utilizado para classificação de documentos. A idéia básica é usar a junção das probabilidades das palavras e categorias para estimar as probabilidades das categorias de um novo documento. Desta maneira, o algoritmo calcula a probabilidade de um documento pertencer a classes diferentes e o atribui à classe cuja probabilidade é a mais alta. A probabilidade é calculada usando-se a regra de *Bayes* e o conjunto de teste é atribuído à classe com a mais alta probabilidade. A parte ingênua (*naive*) do algoritmo NB é a suposição de independência das características da palavra, ou seja, é assumido que o efeito das características de uma palavra cuja probabilidade condicional está associada a uma categoria é independente das características de outras palavras daquela categoria. Apesar desta premissa “ingênua” e simplista, este classificador apresenta um excelente desempenho em várias tarefas de classificação.

O algoritmo *naive bayes* apresenta as seguintes vantagens:

- (i) É de fácil implementação, pois o algoritmo é simples;
- (ii) Não requer nenhum procedimento de aprendizagem (as probabilidades são estimadas baseadas nas frequências de termos);
- (iii) Capacidade de aprender através de exemplos;
- (iv) O processo de classificação é eficiente, já que as características são independentes das outras.

Por outro lado, as seguintes desvantagens deste algoritmo podem ser destacadas:

- (i) Requer o conhecimento inicial de muitas probabilidades;
- (ii) O custo computacional é linear, porém está vinculado à quantidade de palavras e de características existentes.

O método *naive bayes*, inicialmente, pode parecer complexo mas, de fato, possui uma estrutura linear, apesar de possuir uma exponencial em sua estrutura (Equação 4), de acordo com

$$\Pr(C | x) = \frac{1}{\Pr(x)} \exp(\sum_j w_j x_j + b) \quad (4)$$

sendo C a categoria procurada, x uma palavra que compõe o dicionário, w uma palavra que compõe um documento e b é uma constante.

- | |
|--|
| <ol style="list-style-type: none">1. Para cada classe:2. Calcular sua probabilidade3. Para cada atributo do documento:4. Calcular sua probabilidade para cada classe5. Multiplicar todos os valores calculados para esta classe6. Multiplicar o valor obtido pela probabilidade da classe |
|--|

Figura 10 - Pseudocódigo do classificador *Naive Bayes*

3.2.10.2 Classificador de Ranqueamento Linear

De maneira a se obter uma boa *performance* na atividade de categorização/predição, é muitas vezes necessário a criação de vetores de características com dimensões muito elevadas. Embora muitas dessas características não sejam utilizadas, torna-se difícil para uma pessoa dizer quais delas são úteis ou não. Deste modo, os algoritmos de categorização devem ser capazes de manipular um grande conjunto de características e selecionar somente os que forem realmente relevantes. Uma maneira muito utilizada para realizar esta tarefa é pelo uso de algoritmos de ranqueamento linear.

O algoritmo *Naive Bayes*, descrito acima (Figura 10), pode ser considerado um caso especial de um algoritmo de ranqueamento linear. Todavia, sua *performance* pode ser melhorada significativamente usando-se métodos de treinamento mais sofisticados para se obter o vetor de pesos $w = [w_j]$ e o bias b .

Considere o problema de classificação para duas classes. O método de ranqueamento linear consiste em assinalar com uma pontuação positiva as classes identificadas como positivas e com uma pontuação negativa as classes identificadas como negativas. Na Tabela 4 é ilustrado um exemplo do uso do conjunto de pesos para determinar uma pontuação para um documento. Na Tabela 5 é ilustrado um exemplo de cálculo do ranqueamento para um novo documento baseado nos pesos da tabela anterior.

Tabela 4 - Pesos das características para o modelo linear

<i>Palavra</i>	<i>Peso</i>
Dividendos	0,8741
Passos	0,4988
Oito	-0,0866
Pagamento	0,6141
Meses	-0,1801
Divisão	0,9050

Tabela 5 - Cálculo do ranqueamento baseado nos pesos das características

<i>Palavra</i>	<i>Peso(Total)</i>
Dividendos, pagamento, meses	1.3081

Para todas as palavras que ocorrem no documento, encontra-se seu peso correspondente. Esses pesos são, enfim, somados para se determinar a pontuação do documento. Matematicamente, este método é uma função de ranqueamento linear, cuja fórmula geral é dada por:

$$Score(D) = \sum_j w_j x_j + b = wx + b \quad (5)$$

onde D é o documento e w_j é o peso para a j -ésima palavra no dicionário, b é uma constante, e x_j é um ou zero, dependendo da presença ou ausência da j -ésima palavra no documento. Métodos de ranqueamento linear são abordagens clássicas para a

resolução de problemas de categorização/predição. Geometricamente, este método consiste na geração de uma reta ou hiperplano no espaço. Embora superfícies complexas não possam ser ajustadas por uma reta, frequentemente é possível criar características não lineares adequadas, de tal modo que a curva no espaço original esteja sobre um hiperplano no espaço aumentado, com as características não lineares adicionais. Desta maneira, a não linearidade pode ser explicitamente capturada com a construção de sofisticadas características não lineares. Uma vantagem dessa abordagem é que o aspecto da modelagem torna-se muito simples, uma vez que o foco está na criação de características úteis e permitir que o algoritmo possa determinar um peso para cada característica criada. Outra vantagem é que este método funciona de maneira muito eficiente com dados esparsos. Isto é muito importante para aplicações de mineração de textos, uma vez que apesar dos vetores de características apresentarem grandes dimensões, eles são, usualmente, muito esparsos.

3.2.11 Avaliação dos Resultados

A partir de um procedimento de mineração, é necessário avaliar o resultado retornado em função das informações relevantes ou irrelevantes obtidas. Nesta fase, os resultados podem ser analisados com a finalidade de constatar se o objetivo foi alcançado. As medidas de avaliação do desempenho de um sistema são originárias da área de RI e baseadas na idéia de relevância, ou seja, se um documento atender a necessidade de informação do usuário, ele é considerado relevante à solicitação do mesmo. Infelizmente, não é possível evitar o retorno de dados indesejados, porém é possível avaliar se o retorno da busca foi satisfatório, através das métricas: “Precisão”, “Abrangência” e “Medida F”. São estas:

- (i) *Abrangência, Recordação ou Recall*: consiste na avaliação da habilidade do sistema em recuperar os documentos mais relevantes do usuário, medindo a quantidade de itens recuperados, dentre os relevantes na base de dados. A abrangência é dada por:

$$\text{Abrangência} = \frac{\textit{itens_recuperados_relevantes}}{\textit{itens_relevantes}} \quad (6)$$

- (ii) *Precisão ou Precision*: avalia a habilidade do sistema manter os documentos irrelevantes fora do resultado de uma consulta. A precisão é definida pela razão entre a quantidade de itens recuperados dentre os relevantes e o número total de itens recuperados, isto é,

$$\text{Precisão} = \frac{\textit{itens_recuperados_relevantes}}{\textit{total_itens_recuperados}} \quad (7)$$

contemplar separadamente as medidas de *precisão* e *abrangência* pode levar a uma avaliação equivocada do sistema, visto que, geralmente, quando se aumenta a *precisão*, a *abrangência* do sistema é diminuída.

- (iii) *Medida F ou F-Measure*: combina os valores de *precisão* e *abrangência*, de maneira a se obter o desempenho geral do sistema, permitindo um balanceamento entre os dois valores. É definida por:

$$F = \frac{2}{(1/P + 1/C)} \quad \text{ou} \quad F = \frac{(\beta^2 + 1)PC}{\beta^2(P + C)} \quad (8)$$

onde β é o parâmetro que permite a atribuição de diferentes pesos para as medidas de *Precisão* (P) e *Abrangência* (C), sendo 1 o valor geralmente adotado. O valor F é maximizado quando P e C são iguais ou muito próximos, sendo F o ponto de equilíbrio do sistema.

Capítulo 4

Ferramenta de Mineração de Textos - Aîuri

A ferramenta de mineração de textos utilizada no presente trabalho é o sistema Aîuri [17], desenvolvido com base nos conceitos utilizados em aprendizado de máquina. A técnica de categorização automática de textos deste sistema consiste fundamentalmente em:

- i. Disponibilizar um conjunto de documentos pré-classificados nas diversas categorias de interesse;
- ii. Transformar a informação textual contida nos documentos para um formato que possa ser manipulado computacionalmente pelos algoritmos de classificação;
- iii. Escolher os termos mais relevantes do conjunto de documentos, isto é, os que permitem melhor discriminação entre as categorias temáticas sob estudo;
- iv. Definir os pesos para os termos dos documentos, de maneira a possibilitar uma maior discriminação entre as categorias consideradas;
- v. Estimar o modelo de classificação;
- vi. Avaliar a efetividade do modelo estimado.

O sistema Aîuri desenvolvido para um trabalho de Mestrado desta instituição, é segundo seu autor, Anddre Serpa [17], um sistema cooperativo acadêmico de alto desempenho - portal *web* - desenvolvido na linguagem de programação Java, que contém toda uma infra-estrutura para capacitá-lo a executar algoritmos em modo local ou em ambientes de *grids* computacionais. Estes algoritmos são capazes de determinar a qual classe um novo texto pertence baseado no conhecimento obtido de textos anteriores.

Dentre as funcionalidades implementadas, podem ser citadas:

- i. *Carregamento* de arquivos e certificados de usuários;
- ii. Geração de conjuntos de textos para treinamento e testes dos algoritmos;
- iii. Geração de métricas dos modelos gerados;
- iv. Geração dos modelos bayesiano e de categorização linear;
- v. Visualização do status dos trabalhos submetidos.

4.1 Funcionamento do Sistema Aîuri

Segundo Serpa [17], o sistema Aîuri pode ser operado de três formas. A primeira delas, e a mais simples, é o modo “Local”, onde o usuário poderá submeter seus experimentos na própria máquina do portal, sem a necessidade de possuir um certificado para utilização dos serviços de *grid*. Nas outras duas (NACAD ou EELA forma, o usuário deverá, após o *login*, fazer o carregamento de seu certificado para poder operar em um ambiente de *grid* computacional.

O objetivo do portal Aîuri é possibilitar que os usuários executem seus experimentos em ambientes diversificados [17].

O sistema deve ser iniciado através do *browser*, O usuário deve informar sua identificação e senha, bem como selecionar em qual ambiente deseja submeter seus experimentos. Após efetuado o *login*, apresenta-se o menu principal, mostrado na Figura 11.

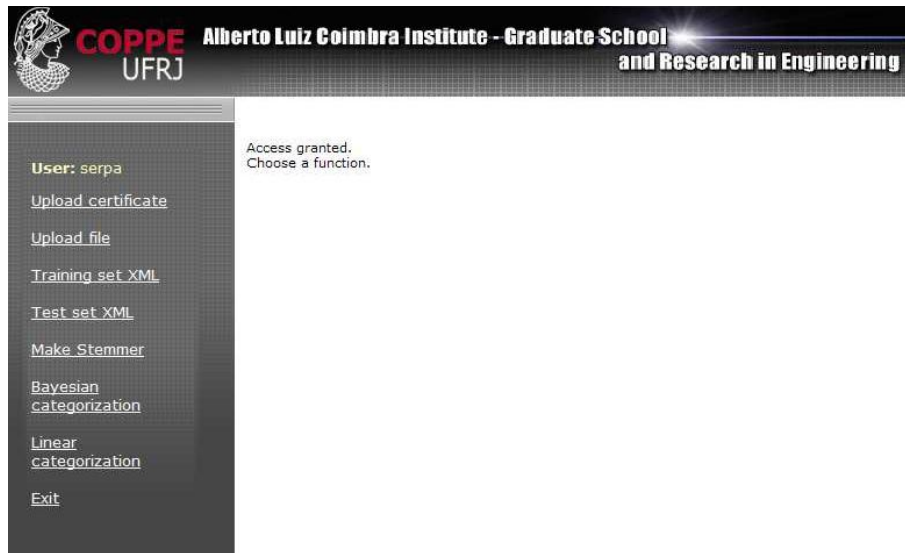


Figura 11 - Menu principal do Sistema Aûri

Todos os arquivos necessários para o processamento, seja do algoritmo bayesiano, seja do algoritmo de ranqueamento linear, devem estar no diretório do usuário na máquina do portal. O usuário deve escolher o menu “*Upload file*”, selecionar o arquivo e proceder o carregamento. Caso os arquivos estejam compactados, o sistema realiza a descompactação automaticamente.

Após o carregamento dos arquivos, devem ser gerados os arquivos de treinamento e teste, que estão no formato “xml”, que contém todos os textos selecionados para treinar e testar o categorizador. O usuário, também, pode realizar o carregamento destes arquivos, sem a necessidade de realizar os procedimentos de geração.

A técnica de *case folding* foi implementada neste sistema. E, segundo Lopes [44], *case folding* consiste na conversão dos dados textuais para o mesmo tipo de

caracter, ou seja, maiúsculo ou minúsculo. No presente trabalho o sistema converte todos os caracteres para minúsculo.

A limpeza consiste na verificação, caracter a caracter, de elementos indesejáveis nos textos, fazendo uma comparação com elementos de um vetor de caracteres previamente determinado. Para isso, foi utilizado um algoritmo que realiza uma pesquisa binária neste vetor.

A pesquisa ou busca binária é um algoritmo de busca em vetores que requer acesso aleatório aos elementos do mesmo. Ela parte do pressuposto de que o vetor está ordenado e realiza sucessivas divisões do espaço de busca comparando o elemento buscado (chave) com o elemento no meio do vetor. Se o elemento do meio do vetor for a chave, a busca termina com sucesso. Caso contrário, se o elemento do meio vier antes do elemento buscado, então a busca continua na metade posterior do vetor. E finalmente, se o elemento do meio vier depois da chave, a busca continua na metade anterior do vetor. A complexidade desse algoritmo é da ordem de $\log_2 n$, onde n é o tamanho do vetor de busca.

Cada caracter é concatenado a uma saída no formato *string*. A soma de todas estas saídas é o texto final, livre dos caracteres não desejáveis. Esta verificação ocorre no momento da conversão dos textos para o formato *XML*, tanto na geração da base de treinamento como da base de teste, garantindo a integridade dos mesmos.

Após a geração dos arquivos de treinamento, teste e, opcionalmente, do arquivo de *stems*, é possível realizar a configuração das propriedades para a submissão do experimento. Na Figura 12 é ilustrada a interface para a configuração das propriedades do algoritmo bayesiano.

Figura 12 - Interface de propriedades para o algoritmo bayesiano

Esta interface possibilita a inclusão de várias propriedades inerentes ao algoritmo em questão, como também em qual recurso disponível este será submetido. A lista dos recursos disponíveis é obtida no momento em que o usuário faz seu *login*.

Para cada recurso selecionado, deve ser configurado seu respectivo conjunto de propriedades. Desta maneira, no momento da submissão, os elementos informados nas propriedades serão enviados para o recurso solicitado.

Após a submissão, alguns arquivos são gerados no recurso e copiados para a área do usuário no portal, para posterior *download*. Os arquivos são identificados com um nome composto por seu tipo (pos, neg e métrica) e o recurso em que foi submetido. O arquivo denominado “pos” contém os textos classificados positivamente, de acordo

com a propriedade “*keyword*” informada. O arquivo “neg” contém o conjunto de textos classificados negativamente, também de acordo com a “*keyword*”. Já o arquivo “métrica” contém as medidas “*precision*”, “*recall*” e “*f-measure*” obtidas quando da execução do algoritmo. A propósito, as métricas de retorno, assim como os arquivos com as classificações podem ser baixados para posterior utilização.

Após a configuração das propriedades, devem ser gerados o dicionário, os vetores de treino e teste, um dos modelos selecionados e, finalmente, o teste do modelo, gerando os respectivos arquivos de saída, que são disponibilizados para o usuário.

A grande facilidade proporcionada pelo portal Aîuri, consiste no encapsulamento das diversas atividades relativas à geração dos modelos bayesiano e de ranqueamento linear. Como já mencionado anteriormente, o sistema também possui funcionalidades da maior importância para as atividades de pré-processamento, como a geração dos arquivos contendo os textos de treinamento e teste e a implementação do arquivo com os *stems*, caso o usuário julgue necessário.

Todavia, as funcionalidades implementadas no portal para as atividades de extração de padrões, mais especificamente, com a utilização das técnicas de categorização, possuem um diferencial, uma vez que encapsulam as tarefas necessárias à geração do modelo, de maneira que fiquem transparentes ao usuário.

4.2 Outras ferramentas de Mineração de Textos

As tarefas de mineração de textos já podem contar com vários tipos de ferramentas computacionais, sejam elas gratuitas ou comerciais.

O uso destas ferramentas é crucial para as atividades inerentes ao processo de mineração de textos. A grande variedade destas ferramentas permite que

pesquisadores e organizações selecionem as que melhor atendem os seus objetivos. Para um estudo sobre uma grande variedade de ferramentas para mineração de textos.

A seguir são descritas algumas destas ferramentas com suas características e aplicabilidade.

4.2.1 Ferramentas de Domínio Público

4.3.1.1 *Text Mine*

Text Mine é um conjunto de ferramentas de mineração de textos escritas em *Perl*, que é uma linguagem apropriada para o desenvolvimento de scripts para servidores *Web*.

Os requisitos básicos desta ferramenta são: (1) linguagem de programação *Perl*; (2) *Apache Server* como servidor de aplicações; (3) banco de dados *MySQL*.

Text Mine disponibiliza as seguintes funcionalidades:

- (i) Busca: a busca pode ser feita a partir de URL's, onde o *crawler* visita as páginas, construindo o conjunto de informações necessárias para as outras funcionalidades, ou em documentos locais à máquina;
- (ii) Extração de informação: é capaz de identificar entidades no texto como nomes, locais e organizações. Para isso, o usuário deve customizar um dicionário de entidades que serão identificadas;
- (iii) *Clustering*: retorna uma coleção de grupos, onde cada grupo é composto por um número variável de documentos similares. Gera uma matriz de similaridades da coleção de documentos e utiliza algoritmo genético para agrupá-los segundo o grau de similaridade existente entre eles;

- (iv) *Tracking*: consiste em organizar as mensagens de texto em categorias;
- (v) Sumarização: consiste em resumir artigos ou documentos da *Web*, extraíndo palavras-chave que determinam o significado do documento.

Esta ferramenta possui versões compatíveis com ambiente *MS Windows* e *Linux*.

4.3.1.2 TMSK - Text-Miner Software Kit

O *TMSK* é um pacote de softwares para mineração preditiva de textos. Possui funcionalidades para pré-processar documentos de textos em formato *XML* e disponibiliza implementações para as seguintes tarefas:

- (i) Pré-processamento: implementa funcionalidades de tokenização, *stemming*, criação de dicionário e detecção de fim de sentença;
- (ii) Predição: possui classificadores baseados em regras de decisão, predição usando *Naïve Bayes* e modelos de ranqueamento linear;
- (iii) Recuperação de informação: implementa o conceito de listas invertidas (termos que apontam para os documentos);
- (iv) *Clustering*: realiza o agrupamento de documentos usando o algoritmo *k-means*;
- (v) Extração de informações: identificação de entidades nomeadas (NER).

O *TMSK* funciona em qualquer tipo de *hardware* e é multiplataforma, sendo necessário para o seu funcionamento apenas uma interface de linha de comando e um interpretador Java (*Java Virtual Machine*) na versão 1.3.1 ou superior.

4.3.1.3 RIKTEXT - Rule Induction Kit for Text

O *RIKTEXT* é um pacote de softwares completo para categorização de documentos baseado em regras de decisão.

Seu objetivo é determinar o melhor conjunto de regras para a predição e classificação, onde o melhor conjunto é formado pelo menor número de regras com erro mínimo.

Os dados para este classificador devem estar na forma de tabela, onde cada linha corresponde a um documento e cada coluna corresponde a um termo do dicionário. Cada célula da planilha recebe valores booleanos indicando a presença ou a ausência da palavra no documento, ou a frequência linear do termo (número de vezes que o termo aparece no documento).

O *RIKTEXT* complementa o *TMSK*, disponibilizando métodos para construção e uso de regras para classificação de documentos. O formato dos dados de entrada do *RIKTEXT* é idêntico ao dos métodos de classificação apresentados no *TMSK* e as exigências mínimas para a execução do *RIKTEXT* são as mesmas do *TMSK*.

4.3.1.4 Intext Software

Intext é uma ferramenta de domínio público para análise de textos. Possui várias funcionalidades que permitem analisar o conteúdo de documentos em inglês e alemão.

Intext é considerado um *toolbox* com várias funcionalidades de análise do texto. Pode ser usado para listar *strings* que ocorrem no texto e suas frequências; analisar conteúdos através da busca de padrões no texto, onde a saída é a lista de ocorrências do padrão identificado pelo usuário; listar referências cruzadas; permutar vocábulos.

Os requisitos mínimos de *hardware* e de *software* são: (1) Sistemas operacionais Win9x, WinME, Win2000, WinNT 4.x e Win XP; (2) Memória: 32 *Mbytes* disponível

(no mínimo); (3) Espaço em disco de 5 *Mbytes* (no mínimo); (4) Textos em: inglês e alemão (espanhol em desenvolvimento); (5) Versões para MacOS, Linux e Unix (HP, IBM, Sun) encontram-se, atualmente, em desenvolvimento.

4.3.2 Ferramentas Comerciais

4.3.2.1 *TextSmart*

O *TextSmart* é um *software* que efetua análise qualitativa dos dados, realizando a mineração de textos sobre registros individuais que descrevem um problema do tipo perguntas e respostas. Normalmente, os registros correspondem ao texto referente a perguntas cujas respostas não têm limite de tamanho. Utiliza estatísticas para analisar automaticamente as palavras-chave e agrupá-las dentro de categorias afins. Também faz uso de uma lista de termos excluídos, que podem ser considerados como *stopwords*, assim como uma lista de *alias* ou termos sinônimos e um conjunto de regras para categorizar os textos. Tanto as listas de *stopwords* e *alias* quanto o conjunto de regras são fornecidos pelo usuário, e, a cada término de processamento as listas podem ser revisadas, com o objetivo de se obter maior refinamento na resposta. Arquivos “.txt” são utilizados como entrada de dados.

O *TextSmart* disponibiliza os seguintes métodos de categorização de textos:

(i) Somente agrupamento: este método baseia seus resultados em *clusters* dos termos, que são determinados usando a forma de *clustering* hierárquico (*Furthest Neighbor Clustering*). Primeiro é calculada a similaridade, usando a medida *Jaccard* para cada par de termos. A medida *Jaccard* é baseada na frequência relativa com a qual um par de termos ocorre ao mesmo tempo nas respostas. Para isso, é necessário contabilizar a presença ou ausência dos pares de termos, através das respostas e armazená-la numa tabela para todos os termos;

(ii) Somente frequência dos termos: este método apenas cria categorias que contém um termo, começando com o termo que ocorre mais frequentemente nas respostas, até o número máximo de categorias especificadas pelo usuário, ou até todos os termos serem categorizados. Esta frequência indica apenas se o termo aparece na resposta, e não quantas vezes ele aparece;

(iii) Agrupamento e frequência do termo: este método usa ambos os métodos descritos anteriormente para criar as categorias. Os termos que não fazem parte de nenhuma categoria são atribuídos individualmente a categorias de um único termo.

4.3.2.2 *STATISTICA Text Miner*

O *STATISTICA Text Miner* é uma extensão opcional do *STATISTICA Data Miner* com funcionalidades para seleção de recuperação de texto, pré-processamento e procedimentos analíticos/interpretativos de mineração de textos não-estruturados (incluindo páginas *Web*).

O aplicativo utiliza algoritmos para processamento analítico e interpretativo, a partir da conversão do texto não-estruturado, inclusive páginas *Web*, em uma matriz de termos.

A análise é feita a partir da redução de palavras ao seu próprio radical, da eliminação de termos sem significado (*stopwords*), além de parametrizações lingüísticas. Assim, o aplicativo elimina uma parte significativa do texto a ser analisado, reduzindo o número de termos da matriz de palavras de texto. Uma vez definidas as palavras com real valor para a análise, são aplicados algoritmos de mineração de textos, por meio dos quais derivam-se modelos preditivos para explicar a possibilidade de ocorrência de termos significantes em outros documentos da

mesma natureza.

As características principais do *STATISTICA Text Miner*:

- (i) Contém várias opções de acesso ao texto em formatos diferentes, incluindo TXT, PDF, PostScript, HTML, XML, e na maioria dos formatos do *Microsoft Office* (por exemplo: DOC, RTF);
- (ii) Provê suporte à “*Web-crawling*”, permitindo extrair documentos de páginas *Web*;
- (iii) Inclui *stoplists* e algoritmos de *stemming* para as línguas: dinamarquês, holandês, inglês, francês, alemão, italiano, português, espanhol, sueco etc. As *stoplists* podem ser editadas pelo usuário conforme necessário. O *software* é projetado de modo que o suporte à línguas adicionais possa ser feito com o mínimo de esforço;
- (iv) Efetua a contagem de frequência de todas as palavras nos documentos, que serve de base para todas as análises numéricas subsequentes. Filtros adicionais podem ser aplicados. Por exemplo, cálculo de frequência normalizada dos termos, frequência inversa dos documentos, gerando uma sumarização numérica dos documentos;
- (v) Disponibiliza métodos de análise estatística que são aplicadas sobre os sumários numéricos dos documentos, técnicas de *clustering*, tais como *kmeans* para identificar documentos similares e técnicas de predição que podem ser usadas para relacionar documentos a indicadores de interesse. Por

exemplo, detecção de fraude, diagnósticos médicos, entre outros.

4.3.2.3 SQL Server 2005

O *SQL Server 2000* já disponibiliza algoritmos de classificação e de *clustering* de dados. Na nova versão *SQL Server 2005*, são implementados não só a execução desses algoritmos, como também componentes para análise de textos, podendo ser usados em conjunto com os algoritmos de mineração de dados.

O *SQL Server 2005* disponibiliza o *SQL Server Integration Services (SSIS)* que possui componentes para mineração de textos e profunda integração com os algoritmos de mineração de dados implementados. Os componentes para mineração de textos permitem identificar relacionamentos entre categorias de negócios e dados (palavras e frases). Além disso, a partir da descoberta de termos chave no texto, encontram-se automaticamente termos de interesse.

Com os componentes do *SSIS* o desenvolvedor pode manipular documentos de texto e efetuar a tokenização, que divide os documentos em palavras e frases, inserindo-as no banco de dados. Outro componente existente calcula as frequências dos termos e a frequência inversa do documento. Com os valores das frequências o *SSIS* cria os vetores compostos por pares (termos e pesos), que descrevem o conteúdo de um documento ou a categoria a qual ele pertence. A partir daí, nove algoritmos, incluindo árvores de decisão e de regressão, *clustering*, regressão logística e linear, redes neurais, *naïve bayes*, associação, *clustering* de sequências e séries temporais estão disponíveis para a geração dos modelos e relatórios.

4.3.2.4 LexiQuest Categorize

É uma ferramenta que automatiza o processo de localização dos itens dos documentos ou parte deles em uma taxonomia, em uma ou mais categorias. Utiliza

técnicas de processamento de linguagem natural (NLP), analisando o texto baseado na sua estrutura gramatical e o contexto. Com isso, apresenta grande eficiência no agrupamento de tipos de texto similares.

É capaz de processar textos em diversas línguas (inglês, francês, espanhol, italiano, alemão e holandês). Além disso, utilizando filtros embutidos, a ferramenta efetua a conversão de textos em HTML, PDF, ASCII e XML e formatos Microsoft (.doc, .xls, .ppt) para um formato único e próprio. Textos importados de bancos de dados e de fontes ODBC também podem ser convertidos. A identificação da linguagem escrita é feita usando *ngrams*, que são combinações de palavras ou caracteres de tamanho *n*.

Capítulo 5

Testes e Avaliação dos Resultados

Para avaliação da qualidade dos resultados gerados alguns testes são efetuados utilizando uma ferramenta de Mineração de Textos. Quatro arquivos de sequências de proteínas referentes aos seguintes organismos: Entamoeba histolytica (E-histo); Trypanosoma cruzi; Candida glabrata e Salmonella enterica foram utilizados para a geração dos modelos de categorização e, posteriormente, tais modelos serviram de base para classificação e comparação de outras sequências.

Para ilustrar os resultados dos modelos categorizados, respectivas medidas de desempenho (*f-measure*, *precision*) e tempos de processamento são utilizados gráficos e tabelas.

5.1 Sequências de Proteínas utilizadas

O banco de sequências de proteínas SWISS-PROT, disponível na Internet, serviu como fonte de dados para o presente trabalho. Para testes, foi extraído desta base um conjunto com quatro modelos de sequências de proteínas.

5.2 Descrição do Problema

O problema consiste na utilização de técnicas de mineração de textos (textmining), algoritmos de categorização bayesiana e de categorização linear, para comparar sequências de proteínas e categorizá-las segundo um conjunto de exemplos de cadeias de proteínas previamente classificadas pelos algoritmos em questão.

O fundamento deste trabalho busca, antes de mais nada, avaliar as ferramentas de textmining para categorização de cadeias de caracteres utilizando exemplos reais, no caso sequências de aminoácidos de proteínas. O modelo adotado neste trabalho, vem sendo alvo de vários trabalhos na comunidade científica. Desta forma, a escolha de cadeias de aminoácidos de organismos distintos tem por objetivo contruir conjuntos de treinamento (sequências de aminoácidos do organismo alvo) e conjuntos de teste (sequências de aminoácidos de outros organismos) distintos, de forma a tornar os processos de comparação e categorização mais criteriosos e específicos. Dentre os trabalhos nesta área de estudo, podemos citar: inferência de estruturas de sequências [75], predição e extração de interações entre cadeias de proteínas [68] [69], reconhecimento de padrões de cadeias de caracteres em proteínas e em literaturas biomédica [83] [84] [85], avaliação dos métodos de categorização de textos [77] [78] [79] [80] [81].

5.3 Proposta de Solução

Na utilização dos procedimentos de classificação de textos ou sequências, o modelo deve seguir a premissa de gerar um conjunto de treinamento de cadeias de proteínas rotuladas como exemplos positivos e a amostra de outras proteínas como exemplos negativos. Usando este conjunto de treinamento, treinam-se sequências classificadoras que serão utilizadas para predizer sobre um banco de sequências quais

são correspondentes à classe. E, dessa forma, considerá-las candidatas a regiões de relevante similaridade entre proteínas.

Para um modelo de classificação de textos, a porção texto (ou nome da proteína) de uma proteína é associada a um vetor de características derivado do próprio texto. O vetor representa um espaço multidimensional onde cada dimensão ou característica corresponde a uma palavra específica, neste caso uma subcadeia da sequência previamente particionada em várias subcadeias de tamanhos fixos. Cada palavra w corresponde a um espaço vetor, $\phi_T(w)$, onde o valor do vetor pode ser booleano indicando a presença ou ausência da palavra, ou as palavras (subcadeias) que ocorrem no texto recebem um valor numérico baseado na sua frequência de ocorrências.

Uma cadeia de texto x é mapeada em um vetor que é soma de todos os vetores correspondentes às palavras presentes no texto, $\phi_T(x) = \sum_{w \in x} \phi_T(w)$. Ao final do processo, todos os textos classificados como positivos são coletados e assinalados a sua respectiva classe ou categoria.

Enquanto no caso da porção sequência, ela também é representada como vetor multidimensional de características. As sequências são representadas como uma coleção de subcadeias de um tamanho fixo k (ou k -mers) obtidas, geralmente, pela identificação de sucessivas quebra de linha (ou line feed) de extensão k no texto. O vetor de características de uma sequência contém uma dimensão para cada possível k -mer. Considerando um k -mer a , a imagem de um k -mer em um vetor da sequência, $\phi_S(a)$, tem valor booleano 1 para indicar a sua ocorrência e 0 para indicar as outras dimensões. A imagem de uma sequência x é a soma de todas as imagens k -mers, $\phi_S(x) = \sum_{a \in x} \phi_S(a)$.

Em resumo, o vetor correspondente à sequência é submetido a um categorizador. O categorizador por sua vez determina se a mesma pertence ou não a determinada

categoria que estiver sendo procurada. Ao final do processo, todas as sequências classificadas como positivas são coletadas e assinaladas à sua respectiva classe. Os categorizadores são obtidos usando um algoritmo de aprendizado de máquina que processa um conjunto de vetores de características dos textos para que sejam assinalados à classe correta. Este processo é apresentado pela Figura 13.

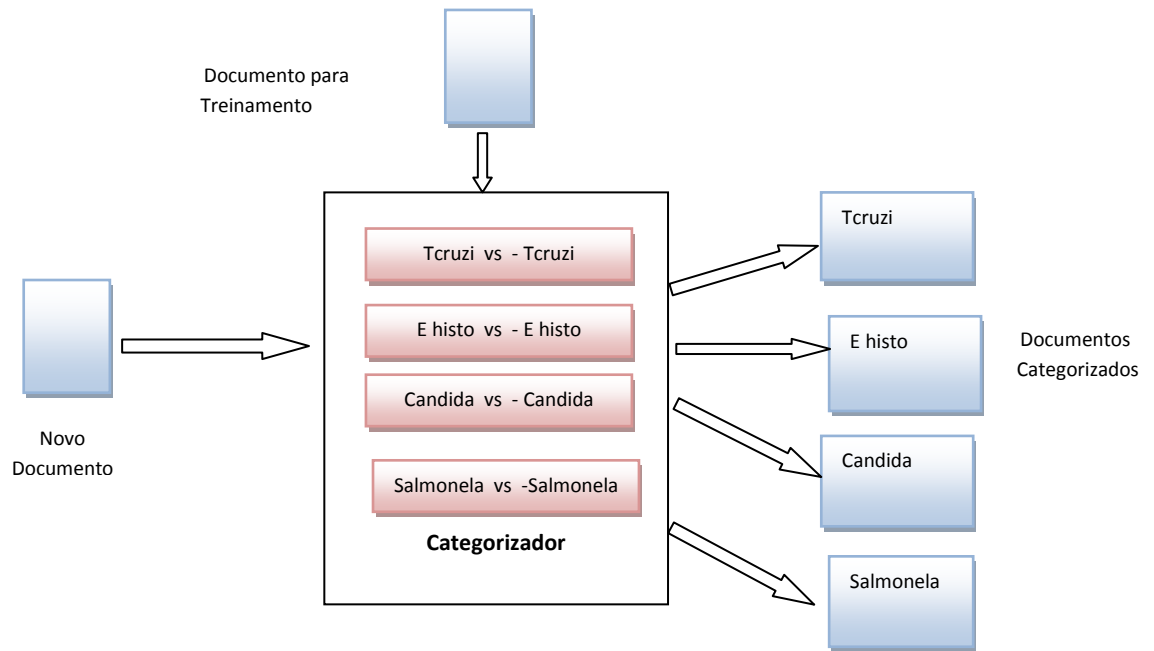


Figura 13 - Modelo de um Categorizador

5.3.1 Predição de Regiões de Similaridades

Dado um conjunto de vetores de treinamento (exemplos positivos e negativos), os classificadores utilizados aprendem uma decisão linear para discriminar entre duas categorias. O resultado é uma classificação linear que pode ser usada para classificar novos conjuntos exemplos para testes.

Supondo que o treinamento consista de vetores de entrada rotulados (x_i, y_i) , $i = 1 \dots m$, onde $x_i \in \mathbb{R}^n$ e $y_i \in \{\pm 1\}$. Pode-se especificar uma regra de classificação linear f para um par (w, b) , onde $w \in \mathbb{R}^n$ e $b \in \mathbb{R}$,

$$f(x) = w \cdot x + b, \quad (9)$$

Onde um ponto x é classificado como positivo (negativo) se $f(x) > 0$ ($f(x) < 0$). Esta é uma regra de classificação que corresponde a uma decisão linear (hiperplano) entre pontos positivos e negativos.

Uma vez que um categorizador é treinado sobre um conjunto de dados, o classificador é representado na sua forma dual como um conjunto de pesos do vetor s_i , um para cada exemplo de treinamento x_i . A forma do classificador é

$$f(x) = \sum_i s_i K(x, x_i) = \sum_i s_i \phi(x) \cdot \phi(x_i) \quad (10)$$

a qual pode ser representada na forma primal como

$$f(x) = \phi(x) \cdot \sum_i s_i \phi(x_i) = \phi(x) \cdot w \quad (11)$$

onde $w = \sum_i s_i \phi(x_i)$,

Por computação explícita $\phi(x_i)$ calcula-se w diretamente. No caso de sequências, este pode ser eficientemente implementado utilizando as mesmas estruturas de dados usadas para processamento do espaço entre sequências. O interesse principal é a porção sequência do vetor de características. Para a porção sequência, w tem um peso para qualquer possível k -mer. A pontuação pode ser interpretada como uma medida do quão discriminativo o k -mer é com respeito ao classificador. Pontuações positivas podem corresponder aos k -mers que tendem a ocorrer no conjunto exemplo ou não em outras proteínas. A pontuação definida para a região da proteína é a soma dos pontos pertinentes aos k -mers contidos na região. Se a pontuação de uma região está acima de um *threshold*, a predição é que tal região é uma região potencialmente funcional associada com o conjunto exemplo de proteínas.

5.4 Descrição do Teste

Para que o processo de mineração de textos seja bem definido, alguns pontos devem ser considerados, tais como: quantidade de textos a ser utilizada para treinamento e teste, características específicas dos textos, classificadores mais adequados, definição das etapas de mineração apropriadas ao tipo de textos usados, etc. Já para a realização dos testes, os dados textuais ficaram divididos em dois conjuntos distintos: de treinamento e de teste. Conforme foi mencionado anteriormente, um conjunto com quatro tipos de organismos foi definido para este trabalho e uma distribuição balanceada destas sequências de proteína para treinamento e teste foi estabelecida em uma relação de 5:1. A Tabela 6 apresenta a distribuição balanceada dos dados.

Tabela 6 - Distribuição balanceada das sequências de proteínas

<i>Tipo de Proteína</i>	<i>Quantidade Treinamento (max.)</i>	<i>Quantidade Teste(max.)</i>
Entamoeba histolytica	500	100
Trypanosoma cruzi	500	100
Salmonella	500	100
Candida	500	100

Os dados foram particionados em quatro grupos e cada grupo contendo sequências pertinentes aos tipos de proteínas identificadas. O quarto grupo foi usado como teste, e os modelos de classificadores foram gerados baseados em cada um dos outros três grupos, sendo gerado um classificador para cada um deles. Os conjuntos de treinamento foram definidos da seguinte forma: o primeiro conjunto de treinamento contém 125 sequências em cada categoria, totalizando 500; o segundo conjunto contém 1000 sequências, com 250 por categoria, e o terceiro com 2000 e respectivas 500 sequências por categoria no último grupo. Cada conjunto possui todos os

textos/sequências do conjunto anterior de modo a tornar os testes homogêneos. O grupo utilizado para teste reúne, em proporções iguais, sequências de proteínas de todas as organismos ou, ainda, apenas de um único organismo diferente do conjunto de sequências utilizado para treinamento.

Para exemplificar a utilização de uma ferramenta de bioinformática na comparação de sequências foi utilizado o programa BLASTP (aplicativo da família de programas BLAST para comparação e classificação de sequências moleculares) sobre os mesmos conjuntos de proteínas previamente definidos. A idéia foi apresentar as funcionalidades desta ferramenta completa baseada em algoritmos de busca de alinhamentos entre cadeais de caracteres e demonstrar que a ferramenta utilizada neste trabalho baseada em algoritmos de categorização de textos poderá, após alguns refinamentos, desenvolver funcionalidades similares, principalmente com a utilização de sistemas de pontuação.

5.4.1 Preparação dos Dados

As sequências são fornecidas em quatro arquivos relativos às proteínas selecionadas, ambos no formato texto (.txt). As sequências foram previamente particionadas em várias subcadeias de tamanho fixo k-mer por uma ferramenta desenvolvida para este trabalho. A ferramenta permite que o parâmetro comprimento k-mer seja alterado conforme a necessidade.

Com o objetivo de tornar este conjunto manipulável pelo portal Aîuri, foi necessária a conversão dos arquivos originais para arquivos individuais com formato ASCII.

Para ambos os conjuntos, os arquivos para treinamento e teste devem obedecer ao seguinte padrão de nomenclatura: **Classe-nome do arquivo.txt**.

Segundo o Serpa [17], as composições Classe e nome-do-arquivo não podem conter acentos. O item “Classe” é obrigatório para os arquivos que compõem o conjunto de treinamento e opcional para o conjunto de teste. A primeira linha de ambos os tipos de arquivos deve conter o título dos textos. Na ausência do título deve ser colocado qualquer caracter do alfabeto. Para estes casos o padrão adotado foi a *string* “xxx”.

Desta maneira, ambos os conjuntos estão prontos para serem convertidos para o formato *XML* e serem utilizados pelo sistema Aûri para a geração dos modelos.

5.4.2 Avaliação dos Resultados dos Categorizadores

Os resultados gerados pelos categorizadores implementados na ferramenta de mineração de texto escolhida com relação às ferramentas tradicionais de comparação de sequências são avaliados em seguida.

A medida utilizada para avaliar o desempenho dos classificadores foi a *f-measure*, que é obtida a partir dos textos do conjunto de teste, já discutida no capítulo 3. Esta medida combina os valores das medidas de *precisão* e *abrangência*, de maneira a se obter o desempenho geral do sistema, permitindo um balanceamento entre os dois valores. A *f-measure* mede a capacidade de generalização do modelo gerado, permitindo verificar se durante o treinamento este assimilou características significativas do conjunto de treinamento que permitem um bom desempenho em outros conjuntos de dados ou se ocorreu *overfitting*, que é um fenômeno no qual o modelo especializa-se nos dados de treinamento. O valor da *f-measure* é maximizado quando a *precisão* ou *abrangência* são iguais ou muito próximas. Deve-se lembrar, que em geral, a eficiência da extração das entidades de um texto se comporta de forma exponencial à complexidade algorítmica. É possível, com heurísticas simples, conforme foi realizado

nos testes que seguem, atingir facilmente um nível de acerto de 68 à 83%. A dificuldade consiste em se obter valores acima de 90%.

Conforme foi apresentado no capítulo 3, o método de comparação de sequências foi baseado no método de programação dinâmica. Algumas extensões dele foram feitas para que o algoritmo se tornasse mais adequado para determinadas tarefas [48]. A complexidade quadrática do algoritmo básico e de suas extensões faz com que eles se tornem inviáveis em determinadas aplicações. Um exemplo importante desse tipo de aplicação é a busca em bases de dados moleculares. Algumas destas bases possuem milhões de sequências. O problema típico dessa aplicação é a comparação de uma dada sequência nova com todas as outras depositadas na base de dados. Isto significa que milhares de comparações precisariam ser feitas [48].

Por este motivo, vários métodos mais rápidos surgiram. Geralmente eles são baseados em heurísticas. Alguns deles são baseados em uma janela de sequências, que é simplesmente um fator de s . A idéia desses métodos é que, se duas sequências são parecidas, então elas terão muitas janelas em comum. Como as técnicas de sequenciamento de nucleotídeos e proteínas têm melhorado muito, e conseqüentemente os repositórios desses dados estão crescendo, a aplicação de busca em bases de dados moleculares tem se tornado a cada dia que passa mais importante. Para auxiliar nessa busca utilizam-se famílias de algoritmos chamadas FAST e BLAST. Os programas dessas famílias estão baseados em heurística que trazem uma melhora significativa nos tempos de respostas das buscas em bases de sequências.

A classificação e a comparação de sequências de proteínas tornam-se cada vez mais importantes na atual revolução biotecnológica, onde cientistas vêm trabalhando, incessantemente, em tecnologias de genômica e proteômica funcional para previsão em larga escala da função de proteínas. No entanto, a classificação funcional é importante

também para o cientista de bancada que realiza análises de individuais ou de pequenos conjuntos de proteínas, ou até mesmo de um simples genoma. Algumas ferramentas estão disponíveis para a classificação de sequências, como pesquisas de similaridades, *motif*-ou programas para reconhecimento de padrões, e assinaturas de proteínas para a identificação de famílias de proteínas e domínios.

Dentre essas ferramentas estão a InterProScan e o CluStr da EMBL(European Molecular Biology Laboratory), a primeira é uma ferramenta que combina vários métodos de reconhecimento de assinaturas de proteínas para alinhar sequências e obter informações adicionais sobre famílias de proteínas. A ferramenta CluSTr oferece uma classificação automática dos grupos de proteínas relacionadas. O agrupamento é baseado na análise de todas as comparações alinhadas entre sequências de proteínas. Ambas trabalham, exclusivamente, sobre as seguintes bases de dados: Pfam, PROSITE, PRINTS, ProDom, SMART, TIGRFAMs, PIRSF, SUPERFAMILY, Gene3D, and PANTHER.

5.4.2.1 Avaliação de Categorizador Bayesiano

Os classificadores bayesianos são classificadores estatísticos, baseados no teorema de *Bayes*, capazes de estimar a qual classe determinado registro de uma amostra pertence. No presente teste, no contexto de mineração de textos, foram utilizados os conceitos do classificador *bayesiano* ingênuo, no qual as variáveis são consideradas independentes entre si. Esta suposição simplifica a abordagem do problema de classificação sem comprometer significativamente a precisão do resultado.

Na análise do experimento, a seguinte abordagem foi definida: geração de um classificador (bayesiano ou linear) sem a utilização de *stopwords* e *stems*, variando somente o número de termos do dicionário. Justifica-se, a não utilização de *stopwords* ou *stems*, pelo fato de sequências de proteínas consistirem basicamente de variadas

combinações de vinte letras, ou aminoácidos, em cadeias de caracteres. Para uma melhor abordagem, consulte o capítulo 2.

Para efeito de avaliação do classificador, os parâmetros *probability-threshold* e *reject-threshold* foram ajustados para diversas combinações, observando-se variações nos resultados. Quando foi atribuído o valor 1 para o parâmetro *reject-threshold*, verificou-se que o classificador obteve *f-measure* nulo, ou seja, todos os textos foram rejeitados, o que também já era esperado.

Sabe-se que um dicionário é construído baseado na coleção de termos existentes no conjunto de treinamento e que seu tamanho é um fator importante. Para esta tarefa, utilizam-se as seguintes técnicas: as *stopwords* que consistem em subtraí-las de palavras sem significado semântico, como artigos, preposições etc., e *stems* que consistem em especializar o dicionário, por meio da extração dos radicais das palavras. Devido a natureza dos dados do presente trabalho, não existe a necessidade do uso destas técnicas sobre cadeias de caracteres.

Fatores como os métodos de predição utilizados e a capacidade computacional, são primordiais para a definição do tamanho do dicionário. Foram realizados testes utilizando dicionários com até 2000 termos (regiões das sequências).

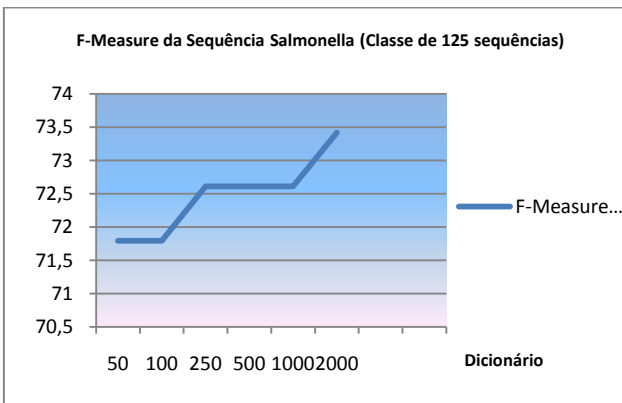
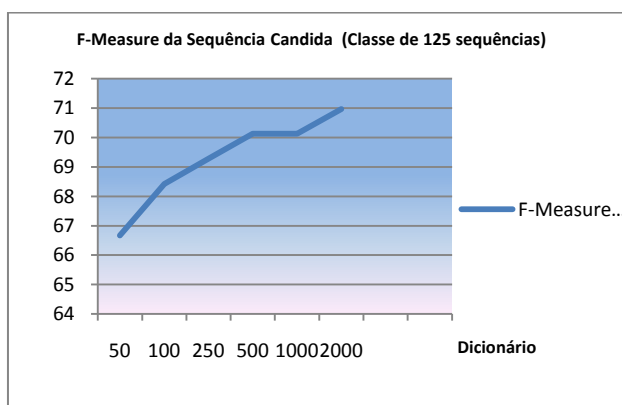
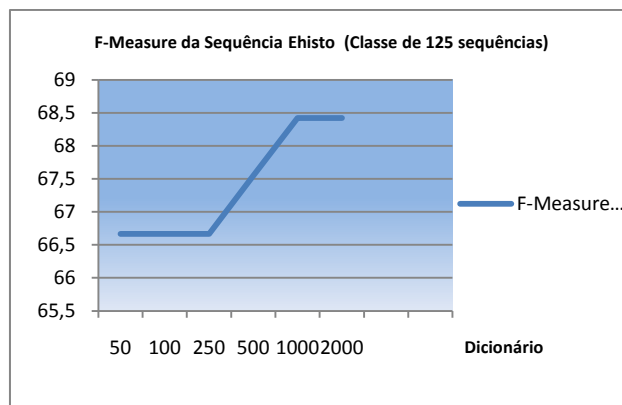
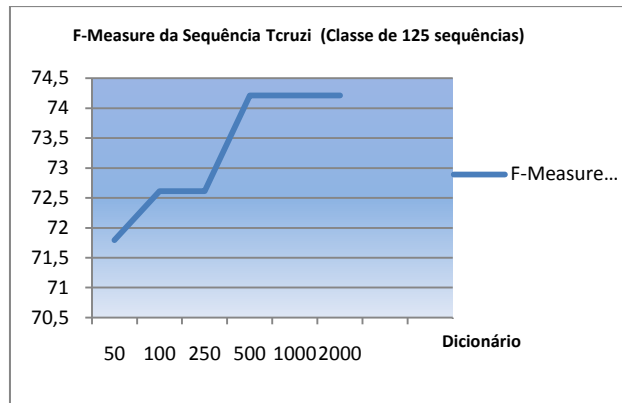


Figura 14 - Resultados do Categorizador Bayesiano para as proteínas do primeiro conjunto de treinamento.

Deve ser observado que dicionários muito grandes tendem a gerar *overfitting*, apresentando baixo desempenho para classificar textos desconhecidos. Dicionários menores, além de evitarem o *overfitting*, reduzem o tempo de processamento dos algoritmos, fato que pode ser observado nos experimentos realizados.

Nos gráficos da Figura 14 são apresentados os resultados obtidos para as classes de proteínas do primeiro conjunto de treinamento com balanceamento na distribuição dos dados (500 sequências, obedecendo a distribuição de 125 sequências por categoria).

Pela observação dos resultados obtidos para a categoria da proteína Tcruzi apresentou os melhores resultados para a *f-measure*. A partir de 500 termos no dicionário os valores obtidos foram equivalentes a 74,21% e com tempo de processamento de 1,5 seg.

Os resultados obtidos para a proteína Salmonella foram próximos aos da Tcruzi, apresentando uma média de 72,47% para a *f-measure* a partir de 500 termos. Enquanto para as categorias Ehisto e Candida, os melhores resultados de *f-measure* foram respectivamente: 68,42% e 71% para 2000 termos no dicionário. Todavia, apresentando uma *f-measure* média em torno de 65%, verificou-se uma maior dificuldade deste classificador em encontrar termos relevantes para estas categorias.

Observa-se que este classificador apresentou padrões similares de crescimento da *f-measure*, para todas as classes, considerando valores acima de 500 termos no dicionário. E que os tempos de processamento obtidos para estas medidas variaram de 1,5 a 2 seg.

Nos gráficos da Figura 15 são apresentados os resultados obtidos para as proteínas do segundo conjunto de treinamento (1000 sequências).

O classificador, para a categoria Salmonella, apresentou o melhor desempenho, com resultados para *f-measure* superiores a 75,31% e dicionários acima de 250 termos.

Para a categoria Tcruzi, o padrão de comportamento apresentou valores para *f-measure* de 72 a 75,39% (para 2000 termos), com significativo crescimento a partir de 500 termos no dicionário. Apesar de serem oriundas de famílias distintas de proteínas, suas resultantes são similares na utilização deste classificador.

O classificador obteve resultados semelhantes para *f-measure* da classe E-histo (média 71%), a partir de 500 termos. Enquanto, que para a classe Candida o melhor desempenho foi alcançado, também, a partir de 500 termos (*f-measure* de 73,41%) no dicionário para uma *f-measure* final de 74,19% para 2000 termos.

Considerando o cenário de treinamento de 250 sequências para cada uma das classes, verificou-se um comportamento similar para ambas, em relação ao valor médio de *f-measure* (73%) apresentado pelas classes para um dicionário de 500 termos ou mais. Quanto aos tempos de processamento, estes ficaram na ordem de 2 seg.

A Figura 16 apresenta os resultados do categorizador para as proteínas considerando o terceiro conjunto de treinamento (2000 sequências).

A categoria Salmonella apresentou um aumento significativo nos resultados de precisão, ou seja, o categorizador obteve êxito na busca termos relevantes para esta categoria. Os valores obtidos para *f-measure* variaram de 78,93% (250 termos) a 82,35% (2000 termos).

Para a categoria E-histo, o categorizador apresentou resultados inferiores aos das outras classes. Verificou-se que o maior valor de *f-measure* foi de 71,49%, o que demonstra que ocorreram dificuldades na recuperação de termos relevantes para esta classe. Sendo assim, esta classe apresenta menor similaridade entre os seus termos (regiões das sequências) e os termos das demais categorias.

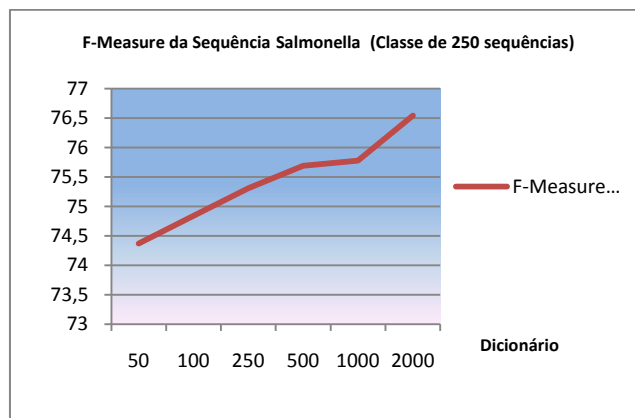
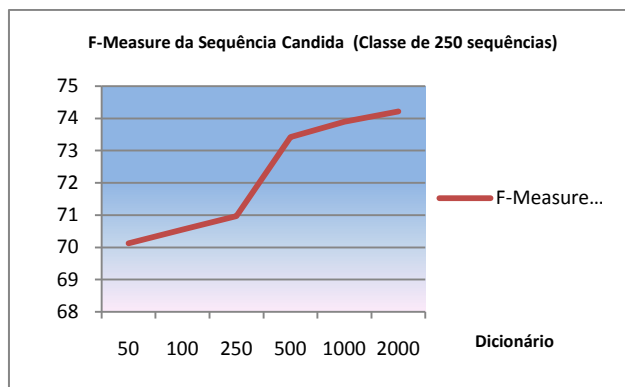
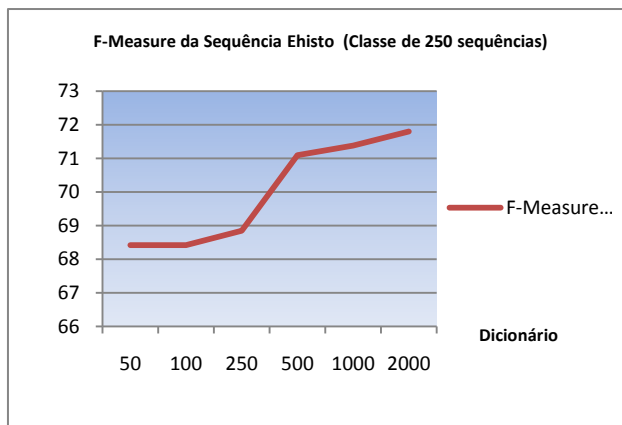
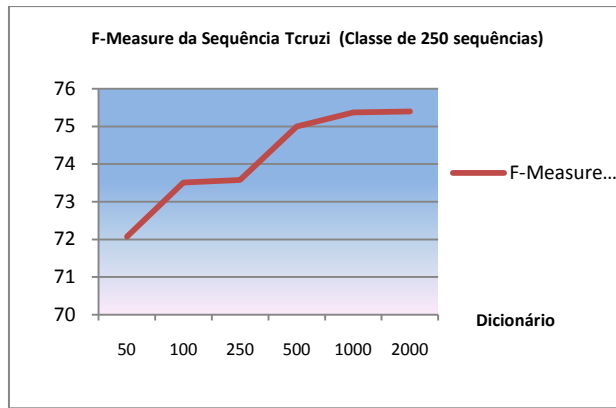


Figura 15- Resultados do Categorizador Bayesiano para as proteínas do segundo conjunto de treinamento.

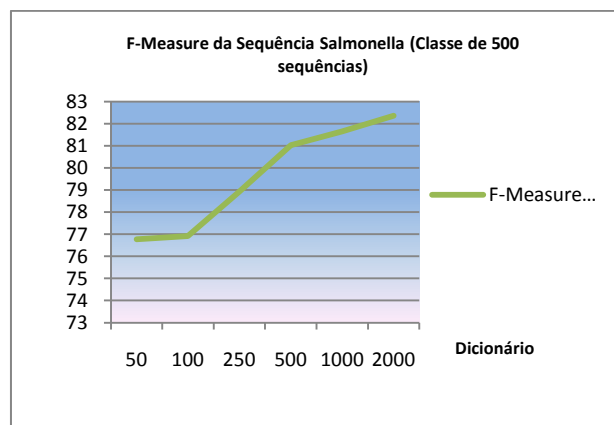
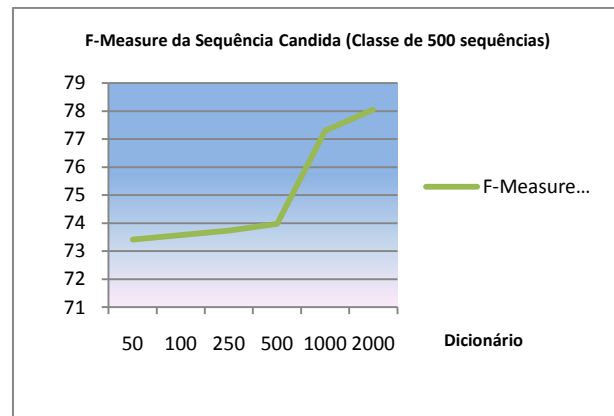
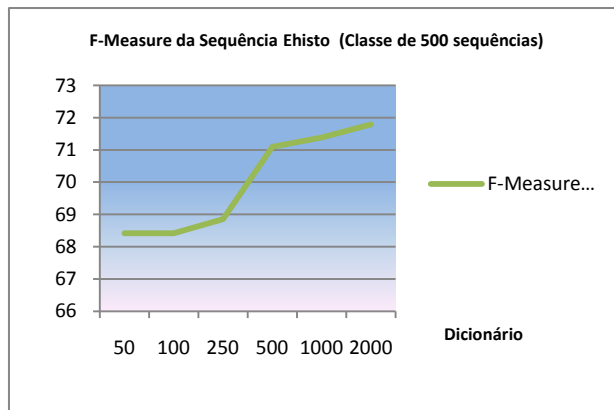
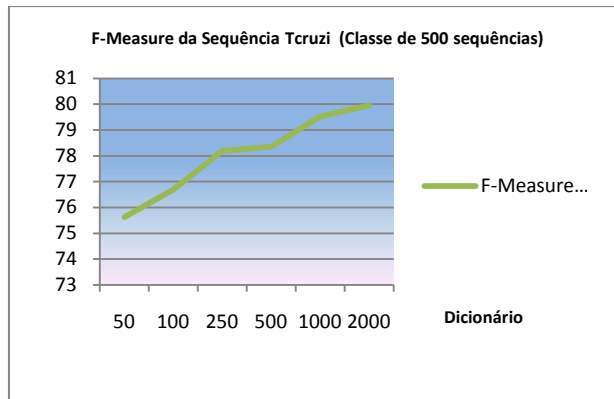


Figura 16 - Resultados do Categorizador Bayesiano para as proteínas do terceiro conjunto de treinamento.

O desempenho do categorizador bayesiano para a categoria Candida obteve melhores resultados a partir de um dicionário com 1000 termos, chegando a alcançar uma *f-measure* de 78% com 2000 termos. Já para a categoria Tcruzi, comparando-se o crescimento da *f-measure* com o crescimento do dicionário, a resultante foi mais homogênea, apresentando os seguintes valores para a métrica: *f-measure* média de 78% e superior de 80% (2000 termos).

Os tempos de processamento aferidos para as categorias do terceiro conjunto atingiram 2 seg.

5.4.2.2 Avaliação de Categorizador de Ranqueamento Linear

Uma vantagem da categorização linear é que a geração do modelo é muito simples, desde que seja feita uma escolha de expressões úteis e que o algoritmo de aprendizado seja capaz de determinar os pesos para cada expressão criada. Os categorizadores baseados em métodos lineares têm abordagem clássica para a solução de problemas de predição.

O modelo para conjuntos de treinamento e teste é similar ao utilizado na categorização Bayesiana.

Para efeito de melhor avaliação dos resultados, as métricas de precisão (*precision*) e recordação (*recall*) foram priorizadas em alguns testes, a partir da manipulação do parâmetro *decision-threshold*.

A priorização da precisão é adequada quando se sabe exatamente qual a classe procurada, com conseqüente retorno de resultados mais precisos. Verificou-se, durante os testes, que a ênfase dada à precisão retornou poucas seqüências, porém com um

maior nível de acertos. Enquanto que a priorização da recordação é mais adequada quando as categorias são desconhecidas, obtendo-se como retorno uma elevada quantidade de textos, mas a maioria fora da sua classificação correta. Para os testes com ênfase na recordação gerou-se um retorno muito maior de documentos, porém com um maior número de classificações erradas.

Em resumo, a relação existente entre as métricas de precisão e recordação define uma contradição. Ou seja, ao maximizar-se a precisão, reduz-se a recordação. E ao maximizar-se a recordação, reduz-se a precisão. Todavia, o principal objetivo é manter os valores elevados para ambas as medidas, priorizando uma ou outra dependendo dos resultados esperados da classificação.

A Figura 17 apresenta os gráficos resultantes do categorizador de ranqueamento linear para as classes de proteínas considerando o primeiro conjunto de treinamento.

O classificador apurou os seguintes valores para a categoria Tcruzi: para 1000 termos, a *f-measure* foi de 75,23% e para 2000, a *f-measure* ficou em 75,77%. Da mesma maneira que no classificador bayesiano, esta categoria atingiu os melhores resultados em relação as outras classes.

Observa-se que para a categoria Salmonella, o classificador obteve o segundo melhor resultado com uma *f-measure* média de 73,87 %. Similar ao resultado do classificador bayesiano, o melhor modelo encontrado para esta classe foi com 2000 termos no dicionário.

Para a categoria Candida, os melhores resultados apurados pelo classificador foram: *f-measure* de 72,53% para 1000 termos e *f-measure* de 72,61% para 2000 termos no dicionário.

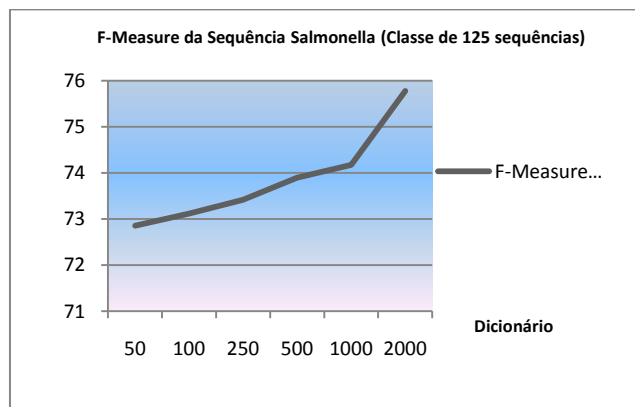
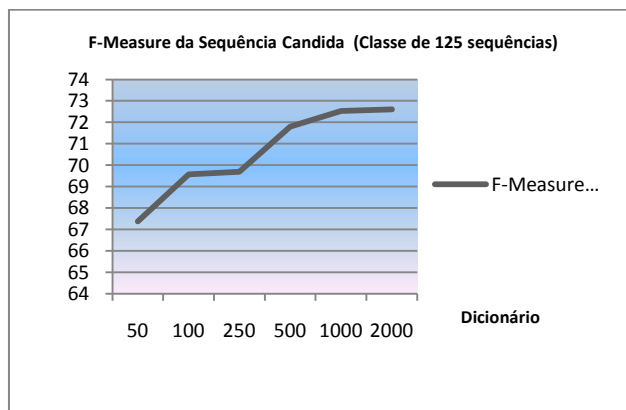
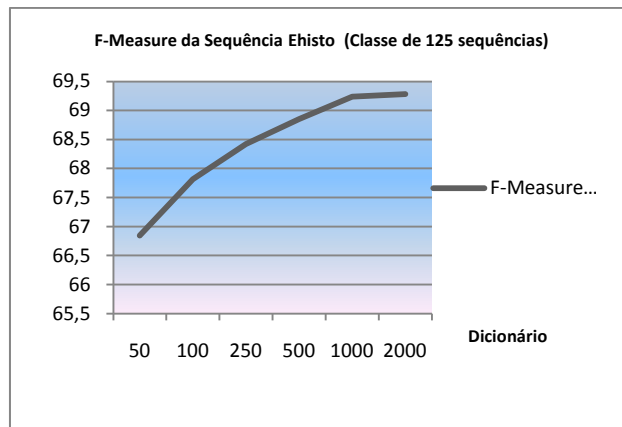
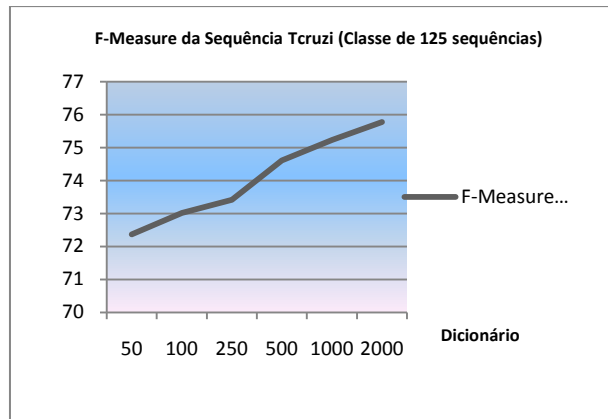


Figura 17 - Resultados do Categorizador de Ranqueamento Linear para as proteínas do primeiro conjunto de treinamento.

Enquanto que para a categoria E-histo, o valor médio da *f-measure* foi equivalente a 68,43% e o melhor modelo encontrado pelo classificador utilizou um dicionário com 2000 termos.

Em geral, o tempo de processamento das classes deste primeiro conjunto variaram em torno de 1,5 a 2 seg.

A Figura 18 apresenta os gráficos resultantes do categorizador de ranqueamento linear para as classes de proteínas considerando o segundo conjunto de treinamento (1000 sequências).

Os melhor resultado obtido pelo categorizador linear para a categoria Salmonella foi com dicionário de 2000 termos, atingindo a *f-measure* de 78,75% no tempo de 2 seg. Tal resultado, demonstra o bom nível de similaridade entre os termos da categoria/classe e do conjunto teste.

Para a categoria Tcruzi, os melhores modelos foram obtidos para dicionário com 1000 termos e 2000 termos, e respectivas *f-measures* de 75,95% e 76,54%. Estas resultantes demonstram boa recuperação de termos, ou seja, existe bom nível de similaridade entre os termos da categoria Tcruzi e o conjunto de teste. O tempo de processamento foi de 2 seg.

O classificador obteve os seguintes resultados para a categoria Candida, a melhor *f-measure* foi de 75% para 2000 termos. Enquanto que para a categoria Ehisto, a *f-measure* obtida, também para 2000 termos, foi de 74,26%. Ambas com tempo de processamento de 2 seg.

As categorias com conjuntos de treinamento de 250 sequências cada (1000 sequências) obtiveram melhor desempenho conforme a variação crescente de termos do dicionário, o que pode indicar existência de *overfitting*. Muito embora, considerando-se

um maior número de termos no dicionário, o tempo de processamento aumenta e a complexidade na recuperação de termos similares, também.

A Figura 19 ilustra os resultados do classificador de ranqueamento linear para as categorias de proteínas considerando o terceiro conjunto de treinamento com 2000 sequências.

Da mesma forma que no classificador bayesiano, com a categoria Salmonella foram obtidos os melhores resultados para todos os experimentos. A utilização de 500 sequências no conjunto de treinamento gerou bons resultados a partir 500 termos no dicionário com *f-measure* máxima igual a 82,95% (para 2000 termos) e tempo de processamento de 2 seg.

Para a classe Tcruzi, a melhor *f-measure* obtida foi de 80,13% para 2000 termos. Enquanto que para a classe Candida a *f-measure* média ficou em torno de 75,11%. O tempo médio de processamento de ambas as classes foi de 2 seg.

Com a categoria Ehisto, em todos os casos, foram obtidos os piores modelos, certamente por esta classe possuir um vocabulário que se diferencia do vocabulário das demais classes. A melhor *f-measure* foi de 72,77% considerando 2000 termos no dicionário.

Verificou-se também que, para a obtenção das melhores medidas para os modelos, os dicionários ficaram em torno de 1000 a 2000 termos, o que é um número elevado que pode gerar *overfitting*.

As médias dos tempos obtidos nos testes podem ser observadas na Tabela 7.

Tabela 7 - Desempenho dos categorizadores

	Bayesiano	Linear
Tempo Máx. de Processamento	2s	2s

Avaliando os testes realizados, verifica-se que os categorizadores (bayesiano e linear) obtiveram um desempenho semelhante. Ambos atingiram medidas de desempenho equivalentes com valores de *f-measure* (abrangência) média acima de 75% e valores de precisão em torno de 65 a 70%, além de serem responsáveis por um um tempo máximo de processamento de 2 seg. E, sendo assim, tornando-se boas opções de categorizadores para a amostra utilizada no experimento, neste caso as cadeias de resíduos das proteínas selecionadas para o trabalho.

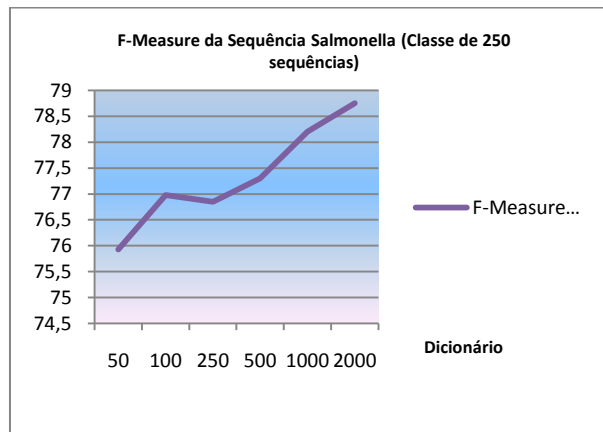
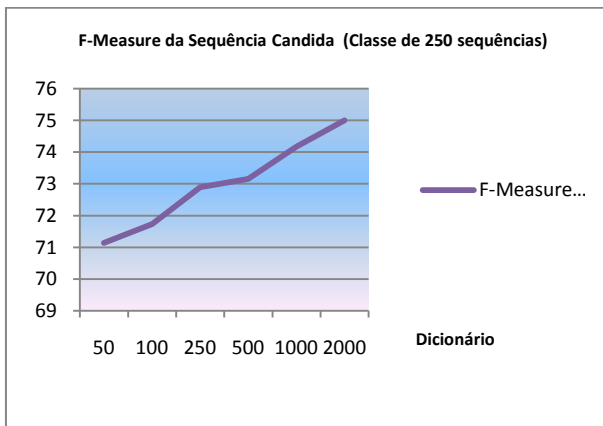
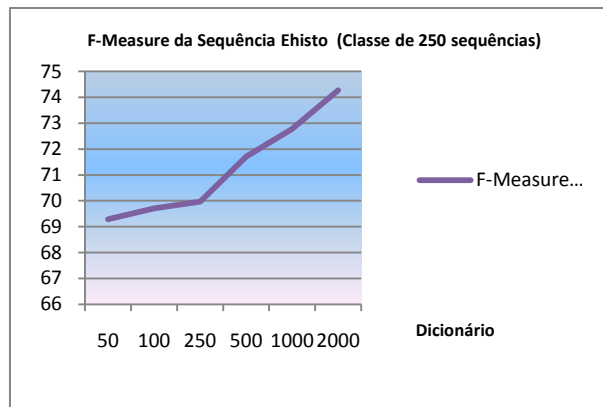
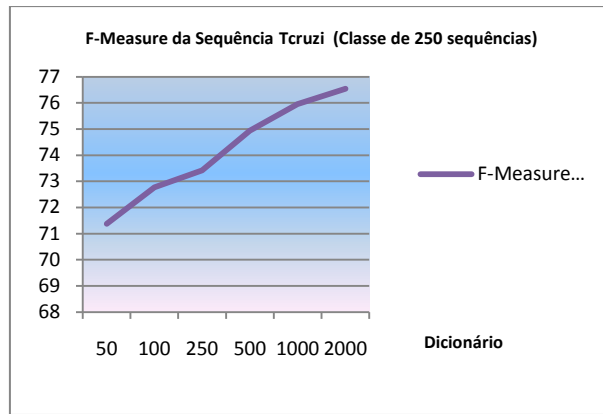


Figura 18 - Resultados do Categorizador de Ranqueamento Linear para as proteínas do segundo conjunto de treinamento.

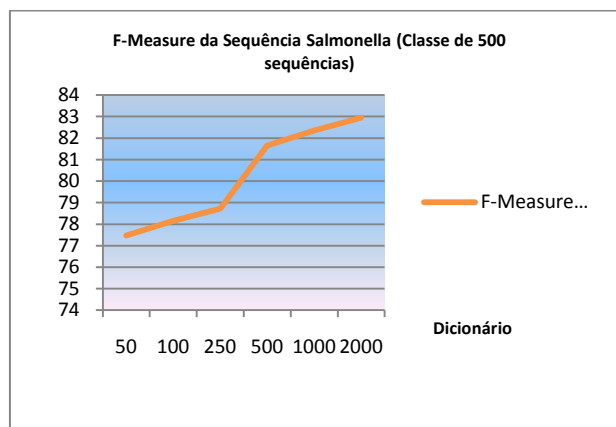
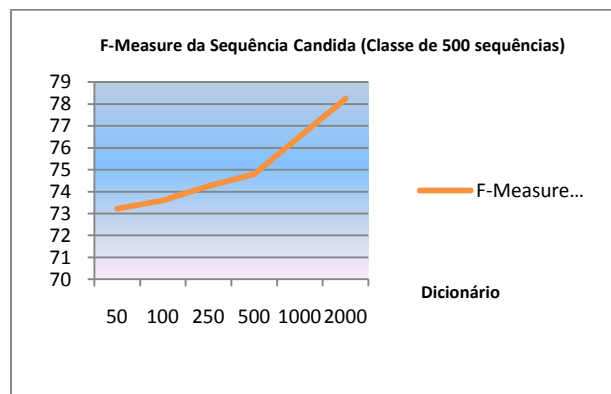
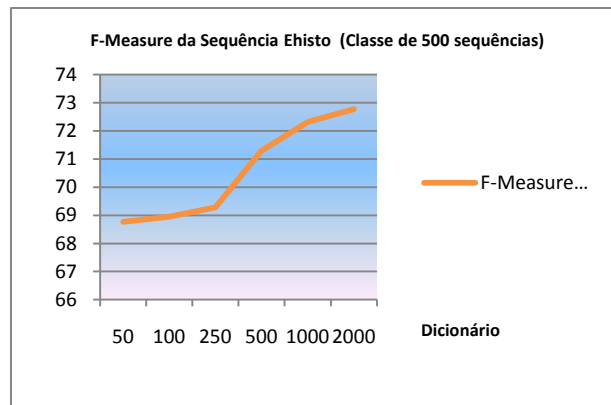
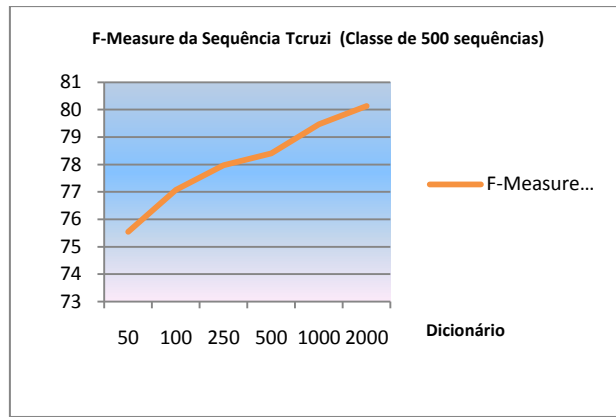


Figura 19 - Resultados do Categorizador de Ranqueamento Linear para as proteínas do terceiro conjunto de treinamento.

5.4.3 Avaliação dos Resultados da Ferramenta BLASTP

Os programas para comparação de sequências geralmente trabalham apenas com três entradas, a sequência a ser comparada, um banco (arquivo) de sequências ou uma segunda sequência para comparação, e o valor do parâmetro *ntupla* (quantidade requerida de resíduos consecutivos para alinhamento) de comparação. Os programas comparam a primeira sequência com todas as outras sequências do arquivo, fornecendo a melhor pontuação por similaridade e alinhamento para cada comparação de par de resíduos. Estes programas calculam um *score* (pontuação) de similaridade local, ou seja, a melhor região de similaridade encontrada na comparação entre duas sequências.

A pontuação da subsequência que apresenta o melhor alinhamento, baseado numa matriz de substituição, é utilizada como a pontuação mais representativa da similaridade entre as duas sequências (a pontuação inicial). Quando se realiza uma pesquisa no banco de sequências, uma pontuação inicial é calculada entre a sequência de consulta e cada uma das sequências do banco. As sequências são ordenadas pela sua pontuação inicial. Além disso, uma pontuação otimizada, a qual permite inserções e exclusões de resíduos, também é computada para as sequências que apresentam as maiores pontuações iniciais de similaridade.

O programa BLASTP busca o melhor alinhamento local ótimo entre duas sequências, atribuindo um *score* (pontuação) de acordo com os critérios utilizados para o caso, como por exemplo: modelo de matriz de substituição (PAM30; BLOSUM62, etc.); penalidades por gaps e outros ajustes.

Conforme foi definido anteriormente, quatro organismos foram definidos para este trabalho (*Entamoeba histolytica*; *Trypanosoma cruzi*; *Candida glabrata* e

Salmonella enterica). Diferentemente da distribuição balanceada destas em conjuntos de treinamento e teste para serem classificadas pelos métodos de categorização bayesiana e linear. No caso da ferramenta de alinhamento BLASTP, uma distribuição, também, balanceada destas sequências foi estabelecida com algumas pequenas diferenças, as quais envolvem a geração de arquivos com sequências a serem comparadas e arquivos com sequências para comparação.

Os dados foram particionados em quatro grupos e cada grupo contendo arquivos de sequências pertinentes às classes de proteínas identificadas. O quarto grupo foi usado como teste e divide as quatro classes de proteínas em arquivos com 25 sequências cada. Ou seja, a mesma proporção utilizada na distribuição das classes no grupo de teste para os categorizadores bayesiano e linear. Os conjuntos de treinamento foram definidos da seguinte forma: o primeiro conjunto de comparação distribui 125 sequências de cada classe em arquivos individuais; o segundo conjunto separa 250 sequências por classe, e o terceiro distribui 500 sequências por classe. Cada conjunto possui todos os textos/sequências do conjunto anterior de modo a tornar os testes homogêneos.

As Figuras 20-a e 20-b apresentam a execução do aplicativo BLASTP na comparação entre as sequências da classe *Trypanosoma cruzi* (arquivo de teste) e as sequências da classe *Salmonella enterica*.



Figura 20-a - Resultados parciais do aplicativo BLASTP para Tcruzi e Salmonella

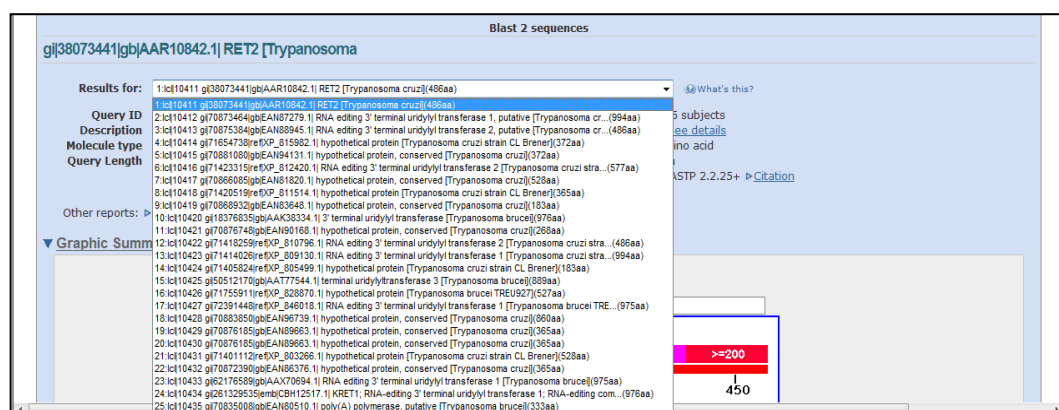


Figura 20-b - Apresentação do conteúdo do arquivo teste da classe Tcruzi

As Figuras 20-c, 20-d e 20-e ilustram três etapas do resultado da execução do programa BLASTP, a saber:

- (i) Distribuição dos alinhamentos locais existentes entre a primeira sequência do arquivo de teste (25 sequências da classe Tcruzi) e o arquivo para comparação (125 sequências da classe Salmonella).
- (ii) As pontuações (*scores*) atribuídas aos alinhamentos obtidos entre as classes de sequências comparadas.
- (iii) Cada alinhamento local obtido e pontuado entre as sequências das classes é representado graficamente.

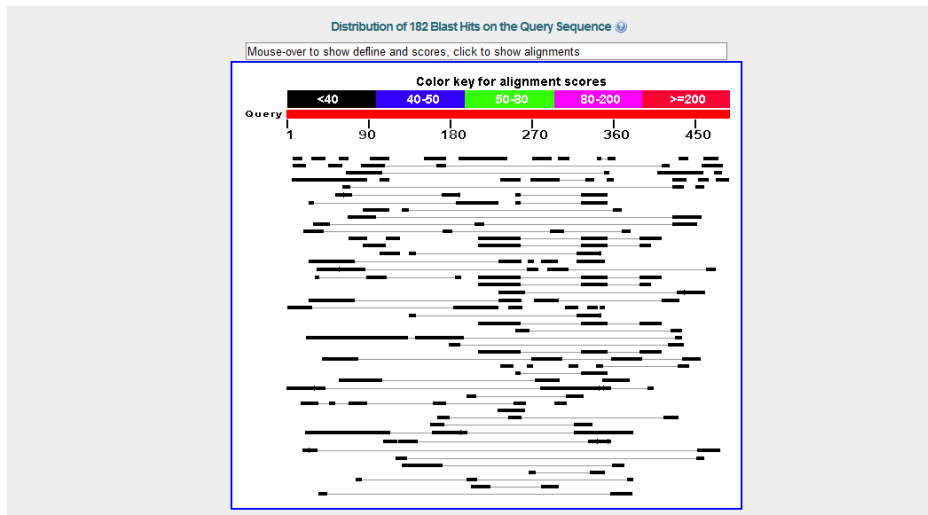


Figura 20-c - Distribuição dos alinhamentos locais existentes entre seqüências

▼ Descriptions
Legend for links to other resources: UniGene GEO Gene Structure Map Viewer PubChem BioAssay

Sequences producing significant alignments:

Accession	Description	Max score	Total score	Query coverage	E value	Links
10437	qi 45758073 gb AAS76285.1 hypothetical protein SCH_001 [Salmon	12.7	12.7	2%	4.2	
10438	qi 45758074 gb AAS76286.1 hypothetical protein SCH_002 [Salmon	16.5	45.0	9%	0.27	
10439	qi 45758075 gb AAS76287.1 hypothetical transmembrane protein [17.7	32.3	9%	0.11	
10440	qi 45758076 gb AAS76288.1 tetracycline efflux protein [Salmonella	17.7	32.3	3%	0.31	
10443	qi 45758079 gb AAS76291.1 transposase [Salmonella enterica subs	18.1	18.1	10%	0.080	
10444	qi 45758080 gb AAS76292.1 transposase [Salmonella enterica subs	16.2	16.2	3%	0.30	
10445	qi 45758081 gb AAS76293.1 resolvase family recombinase [Salmon	13.1	13.1	4%	3.0	
10446	qi 45758082 gb AAS76294.1 putative transposase [Salmonella ente	14.2	14.2	2%	1.4	
10447	qi 45758083 gb AAS76295.1 chloramphenicol acetyl transferase II	16.9	16.9	4%	0.22	
10448	qi 45758084 gb AAS76296.1 hypothetical protein SCH_012 [Salmon	16.8	29.2	4%	0.42	
10449	qi 45758085 gb AAS76297.1 transposase [Salmonella enterica subs	16.5	30.8	7%	0.24	
10450	qi 45758086 gb AAS76298.1 transposase [Salmonella enterica subs	16.5	16.5	4%	0.25	
10452	qi 45758089 gb AAS76301.1 transposase [Salmonella enterica subs	16.5	30.8	7%	0.24	
10453	qi 45758088 gb AAS76300.1 hypothetical protein SCH_017 [Salmon	17.7	17.7	16%	0.12	
10454	qi 45758090 gb AAS76302.1 beta-lactamase [Salmonella enterica :	14.6	28.5	3%	0.97	
10455	qi 45758091 gb AAS76303.1 hypothetical protein SCH_019 [Salmon	11.9	11.9	2%	6.5	
10456	qi 45758092 gb AAS76304.1 mercuric reductase [Salmonella enteri	18.1	65.0	8%	0.091	
10457	qi 45758093 gb AAS76305.1 MerE [Salmonella enterica subsp. ente	16.9	16.9	1%	0.19	
10458	qi 45758094 gb AAS76306.1 MerE [Salmonella enterica subsp. ente	14.6	28.5	12%	1.00	
10459	qi 45758095 gb AAS76307.1 Urf2 [Salmonella enterica subsp. ente	17.3	44.3	11%	0.17	
10460	qi 45758096 gb AAS76308.1 TrnAdelta1 [Salmonella enterica subsp	15.8	58.5	12%	0.45	
10461	qi 45758097 gb AAS76309.1 TnpA [Salmonella enterica subsp. ente	16.2	43.9	20%	0.31	

Figura 20-d - Pontuações (scores) atribuídas aos alinhamentos obtidos entre as classes

Sort alignments for this subject sequence by:
E value Score Percent identity
Query start position Subject start position

Score = 28.5 bits (62), Expect = 7e-05, Method: Compositional matrix adjust.
Identities = 32/113 (28%), Positives = 49/113 (43%), Gaps = 11/113 (10%)

```

Query 23  TTKLNPAPDHVAVANGKAILTENYRVRGPEHMFRTAIRAQQLQGLADKWTPDAKVYCCGS 82
          T+++ P D V   A + E + VG + R A + L+   T A V+ +
Sbjct 28  TSRIQPGSDVIVC---AEMDEQWGVGAFKSRQWLFYAYDLRLK-----TVVAHVPGERT 79

```

Score = 16.9 bits (32), Expect = 0.22, Method: Compositional matrix adjust.
Identities = 11/53 (21%), Positives = 20/53 (38%), Gaps = 0/53 (0%)

```

Query 143  PVVKLRFANDEKVARARYTPLSEEDRKFARTALLDVRNQCIDEREVEYIADKM 195
          P+ + R   V RYT E + R L + + + + + DK+
Sbjct 104  PLYESRLKGLHVISKRYTQRIERHNLNRQLLRLGRKSLSPSKSVLHDRV 156

```

Score = 13.9 bits (24), Expect = 1.7, Method: Compositional matrix adjust.
Identities = 5/7 (71%), Positives = 6/7 (86%), Gaps = 0/7 (0%)

```

Query 428  LNLGRHL 434
          LNL +HL
Sbjct 130  LNLRQHL 136

```

>lcl|10500.gi|45758136|gb|AAS76348.1| hypothetical protein SCH_064 [Salmonella enterica subsp. enterica serovar Choleraesuis str. SC-B67]
Length=213

Sort alignments for this subject sequence by:
E value Score Percent identity

Figura 20-e - Alinhamento local obtido e pontuado entre as seqüências das classes

De acordo com os melhores alinhamentos possíveis obtidos pelo programa BLASTP para os três conjuntos de comparação (125, 250 e 500) para cada classe de proteínas. Os melhores registros de pontuações entre o conjunto teste e os conjuntos de comparação foram verificados para a classe Salmonella. Nesta abordagem, registrou-se uma média de valores para *Score Máximo* em torno de 35% e *Score Total* em torno de 85,2%. Enquanto que as classes *Tripanosoma cruzi* e *Candida* obtiveram os segundos melhores resultados. Sendo que a classe *Entamoeba histolytica* ficou com as pontuações inferiores a estas últimas. Desconsiderando os principais aspectos funcionais de cada ferramenta e considerando apenas a funcionalidade referente a busca de similaridades e entre sequências, observou-se que estes resultados foram semelhantes aos resultados obtidos nas categorizações *bayesiana* e *linear*, as quais baseadas nas medidas de desempenho (*f-measure*, precisão e recordação) geraram modelos que apresentaram, de forma relevante e satisfatória, o nível de similaridade existente entre as quatro categorias de proteínas definidas.

Todavia, os tempos de processamento da ferramenta BLASTP foram aferidos a partir de ambientes remotos, o que inviabiliza uma comparação efetiva e justa (i.e., em igualdade de condições) dos métodos avaliados. Mas, vale ressaltar que, geralmente, o elevado consumo de tempo por algoritmos de comparação de cadeias costuma ser um problema real a ser enfrentado.

Capítulo 6

Conclusões e Trabalhos Futuros

Este capítulo elabora conclusões fundamentadas nos experimentos realizados e descritos no Capítulo 5, além de recomendar novos trabalhos, buscando dar continuidade à linha de pesquisa desta dissertação.

Esta dissertação, busca avaliar as ferramentas de *textmining* para categorização de cadeias de caracteres utilizando exemplos reais, no caso sequências de aminoácidos de proteínas. A metodologia definida utiliza algoritmos de categorização bayesiana e de categorização linear, para comparar sequências de proteínas e categorizá-las segundo um conjunto de exemplos de cadeias de proteínas previamente classificadas pelos algoritmos em questão.

A idéia foi combinar mineração de textos sobre a base de anotações com aprendizagem de texto para classificar proteínas e determinar as regiões similares específicas entre proteínas de diferentes organismos. Levando-se em consideração um pequeno conjunto exemplo de sequências, em comparação a uma quantidade muito maior de texto anotada, ou seja, sequências não rotuladas (classificadas), o modelo definido aprende como o texto está correlacionado com os rótulos e, conjuntamente, aprende sobre sequências e textos, tanto os exemplos (rotuladas) quanto os não

rotulados (anotados). O resultado da aprendizagem é uma sequência classificadora, que pode ser usada para identificar regiões nas proteínas específicas para a classe.

A ferramenta de mineração de textos utilizada no presente trabalho foi o sistema Aîuri [52], desenvolvido com base nos conceitos utilizados em aprendizado de máquina. Dentre as funcionalidades implementadas, podem ser citadas: *Carregamento* de arquivos e certificados de usuários; Geração de conjuntos de textos para treinamento e testes dos algoritmos; Geração de métricas dos modelos gerados; Geração dos modelos bayesiano e de categorização linear; Visualização do status dos trabalhos submetidos.

Para preparação dos dados de entrada foi desenvolvida e implementada uma ferramenta para particionamento das cadeias de aminoácidos dos diferentes organismos em fragmentos de tamanho fixo (k-mer).

A fim de elucidar dúvidas sobre a metodologia empregada, foram descritas as técnicas de classificação implementadas, detalhando os algoritmos bayesiano e de ranqueamento linear, procurando demonstrar seus fundamentos matemáticos, como também realizar um detalhamento sobre as métricas de avaliação utilizadas nesse tipo de processo. Além das técnicas de MT, as técnicas tradicionais de comparação aproximada de cadeias (BLAST) também foram descritas neste trabalho.

6.1 Conclusões

Para os modelos gerados a partir dos métodos de classificação de textos, bayesiana e linear, empregados na comparação e classificação de cadeias moleculares, verificou-se um bom desempenho na maioria dos ensaios aos quais foram submetidos. É importante ressaltar também que, na comparação dos métodos de MT e métodos tradicionais de comparação aproximada de cadeias (dentro das condições de operação

oferecidas pelos ambientes), o único aspecto que deve ser efetivamente considerado é a eficiência das ferramentas na busca aproximada por similaridades e posterior categorização das cadeias. Observando apenas estes aspectos, o método MT (categorizadores Naive Bayes e de ranqueamento linear) ofereceu sempre bons resultados, de fácil interpretação, medidas de abrangência e precisão elevadas e niveladas, além de tempos de execução mínimos. Enquanto que a ferramenta da família BLAST, apesar de reconhecida acurácia e completude na apresentação dos resultados, os quais permitem variadas interpretações teve como principais desvantagens um maior tempo de execução e uma classificação um pouco mais complexa e de difícil interpretação. E como vantagens, a adoção de parâmetros da ferramenta (matriz, penalidades, gaps), os quais, irrefutavelmente, tornam o alinhamento local mais sensível ou mais seletivo, se necessário for.

A partir dos resultados obtidos nos experimentos, demonstrou-se que os métodos de classificação de textos empregados neste trabalho funcionaram adequada e eficientemente ao seu propósito, que era a categorização de cadeias de resíduos (i.e., sequências de proteínas), obtendo boas medidas de desempenho e tempos de processamento reduzidos. Podendo-se assim concluir diante dos experimentos realizados, que os resultados esperados foram atingidos.

6.2 Recomendações para Trabalhos Futuros

A seguir, são relatadas algumas sugestões para a realização de novas pesquisas:

- (i) Implementação de rotina específica de atribuição de pontos aos alinhamentos ocorridos entre sequências moleculares realizada pelos classificadores de texto, visando distribuir pontuações ótimas para as diversas regiões locais de similaridade ocorrentes entre as sequências analisadas.

- (ii) Utilizar os demais recursos do sistema Aîuri como ambiente de execução de *grids* computacionais para classificar sequências moleculares.
- (iii) Desenvolver um método de particionamento eficiente das sequências moleculares em fragmentos de tamanho adequado para a comparação, mas sem comprometimento de sua identificação como sequência válida, o qual beneficiaria a elaboração de planilhas de termos para predição de padrões..

Capítulo 7

Referências Bibliográficas

- [1] ALTSCHUL, S.F., MADDEN, T.L., SCHÄFFER, A.A. et al. **Gapped BLAST and PSI BLAST: a new generation of protein database search programs**. Nucleic Acids Research, [s,1], v.25, n.17, p.3389-3402, jun.jul. 1997.
- [2] ALTSCHUL, Stephen F. **Amino acid substitution matrices from an information theoretic perspective**. Journal of Molecular Biology, v. 219, p. 555-565, fev. 1991
- [3] ALTSCHUL, Stephen F., LIPMAN, David J. **Protein database searches for multiple alignments**. Proc. Natl. Acad. Science, v. 97, p. 5509-5513, jul. 1990.
- [4] ALTSCHUL, Stephen F., MARK, Boguski S., WARREN, Gish, WOOTTON, John C. **Issues in searching molecular sequence databases**. Nature Genetics, v. 6, p. 119-129, fev. 1994.
- [5] APOSTOLIC, A., ATALLAH, J., LARMORE, L., MCFADDIN, S. **Efficient Parallel Algorithms for String Editing and Related Problems**. SIAM J. Comput., v. 19, n. 5, p. 968-988, out. 1990.
- [6] BAEZA-YATES, R., FRAKES, William B.; BAEZA-Yates, Ricardo A. **Information Retrieval: Data Structures & Algorithms**. chapter String Searching Algorithms, pp.219–240. Upper Saddle River, New Jersey, Prentice Hall, 1992.
- [7] BAEZA-YATES, R., RIBEIRO-NETO, B., **Modern Information Retrieval**. Addison-Wesley, 1999.
- [8] BARREL, B. G., ANDERSON, A., SANGER, F. **Different pattern og codon recognition by mammalian mitochondrial tRNAs**. Proc. Natl. Acad. Sci., USA, v. 77, p. 3164-3166, 1980.
- [9] BOYER, R. S.; MOOREM J. S., **A fast string searching**

algorithm. *Comm. ACM* **20** (10): 762–772, 1977.

- [10] CAIRNS, J., STENT, G.S., WATSON, J.D. **Phage and the Origins of Molecular Biology.** New York: Cold Spring Harbor Laboratory, 1996.
- [11] CHEN, M. S., HAN, J., YU, P. S., **Data mining: an overview from a database perspective.** *IEEE Transaction on Knowledge and Data Engineering* v. 8, n. 6, pp.866–883, 1996.
- [12] COLE, R., **Tight Bounds on the Complexity of the Boyer-Moore String Matching Algorithm.** SODA'1991. pp.224~233.
- [13] COLUSSI, L., GALIL, Z., GIANCARLO, R., **On the Exact Complexity of String Matching.** Proceedings of the 31st Annual IEEE Symposium on the Foundations of Computer Science, 1990.
- [14] CORDEIRO, A. D., **Gerador Inteligente de Sistemas com Auto-Aprendizagem para Gestão de Informações e Conhecimento.** Ph.D. dissertation, Universidade Federal de Santa Catarina, Santa Catarina, 2005.
- [15] CRICK, F. H. C., BARNET, L., BRENNER S. J. **General nature of the genetic code for proteins.** *Nature*, v. 192, p. 1227-1232, 1961.
- [16] CRICK, F. **General Nature of the Genetic Code.** Chapter III in *Selected Topics in Modern Biochemistry: Proceedings of the Robert A. Welch Foundation Conferences on Chemical Research. Vol. 8*, edited by W. O. Milligan. Robert A. Welch Foundation Offices, p. 43-65, 1965.
- [17] DA SILVA, A. A. S., **Aiuri: Um Portal para Mineração de Textos Integrado a Grids Computacionais**, 155p. 29,7 cm (COPPE/UFRJ, M.Sc., Engenharia Civil, 2007).
- [18] DA SILVA, C. F., **Uso de informações Linguísticas na etapa de pré-processamento em Mineração de Textos.** Master's Thesis, Universidade do Vale do Rio dos Sinos, São Leopoldo, 2004.
- [19] DAYHOFF, M. **Atlas of Protein Sequence and Structure** (National Biomedical Research Foundation), v. 5, s. 3, 1978.
- [20] DORRE, J., GERSTL, P., SEIKERT, R., **“Text mining: Finding nuggets in mountains of textual data”.** In: *Knowledge Discovery and Data Mining*, pp. 398–401, San Diego, 1999.
- [21] DOS REIS, C. C. T. **Utilização de Métodos de Paralelização em Algoritmos de Pesquisa e Comparação em Sequências Moleculares**, 120p. (NCE/UFRJ, M.Sc.), Rio de Janeiro, 2001.
- [22] EBECKEN, N. F. F., LOPES, M. C. S., COSTA, M. C. A.,

“Sistemas Inteligentes: Fundamentos e Aplicações”. chapter Mineração de Textos. São Paulo, Manole, 2003.

- [23] EDMISTON, E.W., CORE, N., SATZ, J. SMITH, R. **Parallel Processing of biological sequence comparison algorithms**. International Journal of Parallel Programming, v. 17, n. 3, p. 259-275, 1988.
- [24] FAYYAD, U. M., PIATETSKY-SHAPIRO, G., SMYTH, P., **Knowledge discovery and data mining: Towards a unifying framework**. In: *KDD*, pp. 82–88, 1996.
- [25] FELDMAN, R., DAGAN, I., **KDT - knowledge discovery in texts**. In: *Proceedings of the First Int. Conf. on Knowledge Discovery (KDD)*, pp. 112–117, 1995.
- [26] GALIL, Zvi, GIANCARLO, R. **Data structures and algorithms for approximate string matching. J**. Complexity, v. 4, p. 33-72, 1988.
- [27] GALIL, Zvi, GIANCARLO, R. **Improved string matching with k mismatches**. SIGACT News, v. 17, p. 52-54, 1986.
- [28] GALIL, Zvi, PARK, Kunsoo. **An improved algorithm for approximate string matching**. Society for Industrial and Applied Mathematics, v. 19, n. 6, p. 989-999, dez. 1990.
- [29] HARMAN, D., **“How effective is suffixing”**. *Journal of the American Society for Information Science* v. 42, n. 1, pp. 7–15, 1991.
- [30] HEARST, M., **“Untangling text data mining”**. In: *Proceedings of ACL’99: the 37th Annual Meeting of the Association for Computational Linguistics*, pp. 20–26, 1999.
- [31] HENIKOFF, Steven, HENIKOFF, Jorja G. **Amino acid substitution matrices from protein blocks**. Proc. Natl. Acad. Science, v. 89, p. 10915-10919, nov. 1992.
- [32] HUANG, Xiaoqiu. **A space-efficient parallel sequence comparison algorithm for a Message-Passing Multiprocessor**. International Journal of Parallel Programming, v. 18, n. 3, p. 223-239, 1989.
- [33] IVANOV, A. G. **Recognition of na approximate occurrence of words on a Turing machine in real time**. Math, USSR, v. 24, n. 3, p. 479-522, 1985.
- [34] JENSSEN, T.K., A. LAEGREID, J. KOMOROWSKI, AND E. HOVIG, **A literature network of human genes for high-throughput analysis of gene expression**. *Nat Genet*, 28:21-28. 2001.

- [35] KARLIN, S., STEPHEN F. ALTSCHUL, **Methods for assessing the statistical significance of molecular sequence features by using general scoring schemes**. Proc. Natl. Acad. Sci. USA 87:2264-68. 1990.
- [36] KARP, R., RABIN, M., **Efficient randomized pattern-matching algorithms**. IBM Journal of Research and Development 31, 249–260. 1987.
- [37] KNUTH, D.E., MORRIS, J.H., PRATT, V.R., **Fast pattern matching in strings**. Journal of Computing, v. 6, p. 323-350, 1977.
- [38] KROEZE, J. H., MATTHEE, M. C., BOTHMA, T. J. D., **“Differentiating data and text mining terminology”**. In: *SAICSIT '03: Proceedings of the 2003 annual research conference of the South African institute of computer scientists and information technologists on Enablement through technology*, pp. 93–101, Republic of South Africa, 2003. South African Institute for Computer Scientists and Information Technologists. ISBN 1-58113-774-5.
- [39] LANDAU, Gad M., VISHKIN, Uzi., **Fast parallel and serial approximate string matching**. Journal of Algorithms, v. 10, p. 157-169, 1989.
- [40] LENNON, M., PIERCE, D., TARRY, B., WILLET, P., **“An evaluation of some conflation algorithms for information retrieval”**. *Journal of Information Science* v. 3, pp.177–183, 1981.
- [41] LEWIS, D. D., **“Evaluating text categorization”**. In: *Speech and Natural Language Workshop*, pp. 312–318. Morgan Kaufmann, 1991.
- [42] LIPMAN, David J., PEARSON, WILLIAM R., **Fast string matching with k differences**. Journal of Computer and System Sciences. Belgium: Academic Press, v. 37, p. 64-78, ago. 1988.
- [43] LIPMAN, David J., PEARSON, WILLIAM R., **Rapid and sensitive protein similarity searches**. Science, v. 227, p. 1435-1441, 1985.
- [44] LOPES, M. C. S., **Mineração de Dados Textuais utilizando Técnicas de Clustering para o Idioma Português**. Ph.D. dissertation, COPPE/UFRJ, Rio de Janeiro, Tese de Doutorado do Programa de Engenharia Civil da COPPE/UFRJ, 2004.
- [45] LOVINS, J. B., **“Development of a stemming algorithm”**. *Mechanical Translation and Computational Linguistics 11* v. 1, n. 2, pp. 22–31, 1968.
- [46] LUHN, H. P., **The Automatic Creation of Literature Abstracts**. IBM Journal of Research and Development, 2, 157-165. 1958.

- [47] MEADOW, C., BOYCE, B., KRAFT, D., **Text Information Retrieval Systems**. San Diego, Academic Press, 2000.
- [48] MEIDANIS, J., SETUBAL, J. C. **Introduction To Computational Molecular Biology**. 1.ed. [s,l] : IE-Thomson, 1997.
- [49] MOONEY, R. J., NAHM, U. Y., **Text mining with information extraction**. In: *Proceedings of the 4th International MIDP Colloquium*, pp. 141–160. Van Schaik, 2003.
- [50] NEEDLEMAN, S., WUNSCH, C., **A general method applicable to the search for similarities in the amino acid sequence of two proteins**. *Journal of Molecular Biology*, v. 48, n. 443-53, 1970.
- [51] ORENGO, V. M., HUYCK, C. R., **A stemming algorithm for the portuguese language**. In: *Proceedings of the SPIRE Conference. Laguna de San Raphael: [s.n.]*, pp. 13–15, 2001.
- [52] PAICE, C., **An evaluation method for stemming algorithms**. In: *Proceedings Conference on Research and Development in Information Retrieval*, pp. 42–50, London, Springer Verlag, 1994.
- [53] PEARSON, William R., **Rapid and sensitive sequence comparison with FASTP and FASTA**. *Methods in Enzymology*, v. 183, p. 63-98, Academic Press, 1990.
- [54] PELHAM, H. R. B., JACKSON, R. J., **An efficient mRNA dependent traslation system from reticulocytes lysates**. *Eur. J. Biochem.*, v. 67, p. 247-256, Academic Press, 1976.
- [55] PORTER, M. F., **“An algorithm for suffix stripping”**. *Program: electronic library and information systems* v. 14, n. 3, pp. 130–137, 1980.
- [56] RIJSBERGEN, C. V., **Information Retrieval**. 2 ed., London, Butterworths,.147p, 1979.
- [57] RIORDAN, C., SORENSEN, H. **Information filtering and retrieval: An overview**. URL <http://citeseer.nj.nec.com/483228.html>.
- [58] SALTON, G., BUCKLEY, C., **Improving retrieval performance by relevance feedback**. Department of Computer Science-Cornell University, TECHNICAL REPORT, 1987.
- [59] SALTON, G., MACGILL, M., **Introduction to Modern Information Retrieval**. New York, McGRAW-Hill, 1983. 448p.
- [60] SALTON, G., MACGILL, M., **Introduction to Modern Information Retrieval**. New York, McGRAW-Hill, 1984.

- [61] SANGER, F., THOMPSON, E. O. P. **The amino acid sequence in the glycol chain of insulin.** *Biochem. J.*, v. 53, p. 353-374, 1953.
- [62] SHATKAY, H. AND R. FELDMAN, **Mining the bio-medical literature in the genomic era: an overview.** *J Comput Biol*, 10:821-855. 2003.
- [63] SMITH, T.F., WATERMAN, N.S. **Identification of common molecular subsequences.** *J. Mol. Biol.*, 147, 195–197, 1981.
- [64] TAN, A., “**Text mining: The state of the art and the challenges**”. In: *Proceedings of the Pacific Asia Conf on Knowledge Discovery and Data Mining PAKDD’99 workshop on Knowledge Discovery from Advanced Databases*, pp. 65–70, 1999.
- [65] UKKONEN, ESKO. **Algorithms for approximate string matching.** *Information and Control*, v. 64, p. 100-118, 1985.
- [66] VALENCIA, A. AND F. PAZOS . **Computational methods for the prediction of protein interactions.** *Current opinion in structural biology* 12(3): 368-373, 2002.
- [67] V EHLER, F. **Modular Text Mining For Protein-Protein Interactions Extraction.** Thèse de doctorat. UNIVERSITÉ DE GENÈVE. Département d’informatique. FACULTÉ DES SCIENCES. 2009.
- [68] WATERMAN, M.S., EGGERT, M., **A new algorithm for Best subsequence alignments with application to tRNA-tRNA comparisons.** *J. Mol. Biol* 197, 723±728. 1987.
- [69] WATSON, J. D. **The Double Helix.** Atheneum. New York: New American Library, 1969.
- [70] WATSON, J.D., HOPKINS, N., ROBERTS, J., STEITZ, A., WEINER, A. **Molecular Biology of the Gene.** 4 ed., Menlo Park: Califórnia, Benjamin Cummings, 1987.
- [71] WEN, J. AND Z. LI. **Semantic Smoothing the Multinomial Naive Bayes for Biomedical Literature Classification.** 2007 IEEE International Conference on Granular Computing, IEEE Computer Society. 2007.
- [72] WEISS, S. M., INDURKHYA, N., ZHANG, T., DAMERAU, F. J., **Text Mining: Predictive Methods for Analyzing Unstructured Information.** New York, Springer Science Business Media, 237p, 2005.
- [73] WILBUR, W.J., LIPMAN, David J. **Rapid similarity searches of nucleic acid and protein data banks.** *Proc. Natl. Acad. Science*, v. 80,

p. 726-730, fev. 1983.

- [74] WILKINSON, D. AND HUBERMAN, B.A. **A Method for Finding Communities of Related Genes.** *Proc. Natl. Acad. Sci. U.S.A.* 101(Suppl 1): 5241-5248. 2004.
- [75] WITTEN, IAN H. **Adaptive Text Mining: Inferring Structure from Sequences**, Department of Computer Science, University of Waikato, Hamilton, New Zealand. February 2001.
- [76] YANDELL, M. D. AND MAJOROS, W. H., **Genomics and natural language processing.** *Nat. Rev. Genet.*, 3, 601–610. 2002.
- [77] YANG, S. M., X.-B. WU, et al. (2002). **Relative term-frequency based feature selection for text categorization.** *Machine Learning and Cybernetics.* 2001.
- [78] YANG, Y. AND X. LIU (1999). **A re-examination of text categorization methods.** 22nd annual international ACM SIGIR conference on Research and development in information retrieval Berkeley, California, United States. 1999.
- [79] YOON, K. AND S. KWEK (2007). **A data reduction approach for resolving the imbalanced data issue in functional genomics.** *Neural Computing & Applications* 16(3): 295-306. 2007.
- [80] ZHANG, B. (2006). **Intelligent Fusion of Evidence from Multiple Sources for Text Classification.** Computer Science, Virginia Polytechnic Institute and State University. **PhD.** 2006.
- [81] ZHENG, Z. AND R. SRIHARI. **Optimally Combining Positive and Negative Features for Text Categorization.** *ICML Workshop on Learning from Imbalanced Data Sets II*, Washington, DC. 2003.
- [82] ZIPF, G. K., **Human Behavior and the Principle of Least Effort.** Addison-Wesley, 1949.
- [83] ZHOU, G., D. SHEN, et al..**Recognition of protein/gene names from text using an ensemble of classifiers.** *BMC Bioinformatics* 6(1), 2005.
- [84] ZHOU, G. AND J. SU. **Named Entity Recognition using an HMM-based Chunk Tagger.** 40th Annual Meeting on Association for Computational Linguistics, Philadelphia, Pennsylvania. 2001.
- [85] ZHOU, G., J. ZHANG, et al..**Recognizing Names in Biomedical Texts: a Machine Learning Approach.** *Bioinformatics* 20(7), 2004.