ALGORITMOS DE STEMMING E O ESTUDO DE PROTEOMAS

Reinaldo Viana Alvares

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: Rubem Pinto Mondaini

Rio de Janeiro

Janeiro de 2014

ALGORITMOS DE STEMMING E O ESTUDO DE PROTEOMAS

Reinaldo Viana Alvares

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 ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

Examinado por:

_____

Prof. Rubem Pinto Mondaini, D. Sc.

_____

Prof. Álvaro Restuccia Núñez, Ph.D.

_____

Prof. Nicolas Carels, Ph.D.

_____

Prof. Eduardo Massad, D. Sc.

_____

Prof. Leonardo Mondaini, D. Sc.

_____

Prof. Ricardo Cordeiro de Farias, D. Sc.

Rio de Janeiro

Janeiro de 2014

Alvares, Reinaldo Viana

Algoritmos de Stemming e o Estudo de Proteomas / Reinaldo Viana Alvares. – Rio de Janeiro: UFRJ/COPPE, 2014.

X, 72 p.: il.; 29,7 cm.

Orientador: Rubem Pinto Mondaini

Tese (doutorado) – UFRJ/ COPPE/ Programa de Engenharia de Sistemas e Computação, 2014.

Referencias Bibliográficas: p. 40-46.

1. Algoritmos de Stemming. 2. Recuperação de Informação. 3. Biologia Computacional. I. Mondaini, Rubem Pinto. II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia de Sistemas e Computação. III. Título.

DEDICATÓRIA

À minha esposa.

AGRADECIMENTOS

A Deus, que iluminou meu longo caminho, mostrando as melhores soluções para tantos problemas difíceis que se apresentaram durante a realização deste trabalho.

Ao meu orientador Rubem P. Mondaini, pela oportunidade de elaboração e acompanhamento deste trabalho, me proporcionando melhores oportunidades na vida acadêmica e profissional.

À minha esposa Nathielly Campos, por todo seu amor, carinho, apoio e principalmente compreensão em tantos momentos difíceis nesta caminhada tão longa e difícil.

Aos meus pais, Amilcar Ribeiro Alvares e Terezinha de Jesus Viana Alvares, por toda a educação que me proporcionaram.

E a todos que contribuíram direta ou indiretamente para a conclusão deste trabalho.

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

ALGORITMOS DE STEMMING E O ESTUDO DE PROTEOMAS

Reinaldo Viana Alvares

Janeiro/2014

Orientador: Rubem Pinto Mondaini

Programa: Engenharia de Sistemas e Computação

Algoritmos de *stemming* são úteis na área da Recuperação da Informação (RI) na medida em que geram uma representação concisa para palavras que apontem para o mesmo significado base. Podem ser concebidos por meio do uso de uma lista de prefixos e sufixos, ou de forma menos dependente do idioma, tendo como vantagem a concepção de uma solução que possa servir para diversas linguagens.

O estudo de proteínas tem se desenvolvido de forma promissora nos últimos anos, e diversas informações biológicas estão disponíveis na *web*. Há informação útil contida nas estruturas primárias e secundárias das proteínas, que no contexto desta tese são tratadas sob um ponto de vista linguístico (MOTOMURA *et al*., 2013).

Neste trabalho são estudados algoritmos de *stemming* com a proposta de aplicá-los às estruturas de proteínas, buscando entender a interseção entre essas áreas.

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

STEMMING ALGORITHMS AND STUDY IN PROTEOMS

Reinaldo Viana Alvares

January/2014

Advisor: Rubem Pinto Mondaini

Department: Systems and Computer Engineering

Stemming algorithms are useful in the field of information retrieval (IR) as they generate a concise representation for words that point to the same meaning. They can be designed by using a list of prefixes and suffixes or in a language-dependent manner, with the advantage of the solution that can be used for various languages.

The study of proteins has developed promisingly in recent years, and several biological information are available on the web. There is useful information contained in the primary and secondary structures of proteins, which in the context of this thesis are treated under a linguistic point of view (MOTOMURA *et al*., 2013).

Studied in this work are stemming algorithms with the proposal to apply them to protein structures, in order to understand the intersection between these areas.

SUMÁRIO

LISTA DE FIGURAS E TABELAS

Capítulo I – Introdução

1.1 Considerações iniciais sobre *Stemming*

Algoritmos de *stemming* têm sido utilizados em diversas aplicações computacionais presentes no cotidiano da sociedade, na medida em que transformam formas variantes das palavras em uma representação concisa e precisa, comumente chamada *stem*. O objetivo é que tal representação seja genérica o suficiente para capturar a essência das palavras e a de suas diversas variações.

O processo de *stemming* é considerado uma técnica básica presente em vários sistemas de Recuperação de Informação (RI). Tais sistemas são úteis na gerência de informações em virtude do crescimento exponencial do volume das mesmas, o que justifica o desenvolvimento de estudos e pesquisas neste ramo.

No desenvolvimento de algoritmos de *stemming*, dois tipos de erro acontecem com frequência: *overstemming* e *understemming* (STEIN & POTTHAST 2007). O primeiro, quando são removidas mais letras do que o devido, permitindo que palavras com sentidos diferentes apontem para o mesmo *stem*. Por exemplo, o *stem* **comp** para as palavras **computador** e **comparar**. O segundo, quando letras são deixadas a mais, surgindo *stems* diferentes para palavras com mesmo significado. Por exemplo, os *stems* **biolo** e **biolog**, respectivamente, para as palavras **biologia** e **biologista**.

Em geral, um bom *stemmer* – programa computacional que realiza o processo de *stemming* – corresponde a um tipo de solução computacional que, na medida do possível, gera poucos erros de *overstemming* e *understemming*. No entanto, em se tratando do projeto desses algoritmos, a atividade de minimizar um desses erros tem como consequência o aumento do outro e vice-versa.

Convém esclarecer a diferença entre o termo *stemming* - foco deste trabalho - do termo lematização. Lematização refere-se à representação da palavra no masculino singular para adjetivos e substantivos, e o infinitivo para as formas verbais (LUCCA & NUNES 2002). *Stemming* corresponde à retirada de afixos (prefixos e/ou sufixos) da palavra. São termos distintos, embora eventualmente possam ter a mesma forma gráfica.

Um dos principais objetivos dos mecanismos de busca e recuperação de informações na *Web* é que o usuário possa, a partir de argumentos de pesquisa, encontrar a informação desejada. Os principais mecanismos de buscas utilizam, dentre outras soluções, algoritmos de *stemming* para auxiliar nesta tarefa (UYAR, 2009).

O processo de *stemming* pode ser abordado como parte de um contexto maior, chamado Mineração de Texto, que corresponde à atividade de extração de padrões interessantes e não triviais a partir de textos. Trata-se de uma tecnologia que tem sido empregada em várias aplicações, a citar: extração automática de resumos, categorização automática de mensagens de correio eletrônico, entre outras.

Em geral, o processo de Mineração de Texto é dividido em três etapas: preparação de dados, extração de conhecimento e pós-processamento.

A fase de preparação dos dados tem como objetivo tratar o conjunto de dados textuais que servirá como entrada para a fase de extração de conhecimento. Espera-se que este conjunto de dados possa representar a maior quantidade possível de características relevantes dos documentos. Em geral, esta fase é constituída por três etapas:

- uso de *thesaurus*;
- aplicação de *stemming*, e ;
- eliminação de *stop words*.

Um *thesaurus* pode ser definido como um vocabulário controlado que representa sinônimos, hierarquias e relacionamentos associativos entre termos.

A vantagem do processo de *stemming* na etapa de pré-processamento de textos ocorre pelo fato do mesmo identificar similaridades em função da morfologia das palavras, reduzindo o número de atributos do texto, visto que palavras com morfologia semelhante representam de forma genérica o mesmo conceito.

Stop words são palavras que de forma geral não representam informações relevantes para o contexto geral do texto.

Após a etapa de pré-processamento, há a fase de extração de conhecimento, por meio da execução das tarefas de mineração, as quais representam classes de problemas a serem solucionados. Finalmente, a etapa de pós-processamento, a qual ajuda na avaliação das descobertas.

Duas abordagens são comumente utilizadas no projeto de algoritmos de *stemming*: removedores de afixos e *stemmers* estatísticos: a primeira, é dependente das características do idioma o qual o algoritmo é projetado. Normalmente é criada uma lista de prefixos e sufixos e cada palavra é processada, de acordo com critérios estabelecidos para realizar o *melhor* corte. A segunda, independe de prévio conhecimento da estrutura de prefixos e sufixos. Uma ideia simplista seria estabelecer que o *stem* da palavra corresponde aos primeiros n caracteres da mesma.

1.2 Considerações iniciais sobre Biologia Molecular

Em meados do século passado, diversos pesquisadores questionavam a natureza química e física do material genético. Com a descoberta da estrutura do DNA, o qual é responsável pelo armazenamento da informação genética, a descoberta do código genético e do fluxo da informação biológica - DNA / RNA / proteínas - essas macromoléculas ganharam atenção de um ramo interdisciplinar, denominado Biologia Molecular.

Com o surgimento de métodos automáticos para sequenciamento de macromoléculas, houve forte demanda por estruturas de armazenamento desses grandes volumes de dados. Atualmente, a maior parte dos dados está disponível em banco de dados públicos que podem ser acessados via *web*. A Bioinformática surge como meio de auxiliar os pesquisadores no processo de armazenamento, análise e extração de informações úteis dessa vasta quantidade de dados.

Convém ressaltar que o conhecimento necessário para a realização de pesquisas na área é vasto e interdisciplinar. Envolve, por exemplo: Algoritmos, Biologia, Estatística, Física, Matemática, Química, além de diversas disciplinas. Parece que cada vez mais haverá integração entre vários conhecimentos visando análise de dados em Biologia Molecular.

O sequenciamento de proteínas ocorrido nos últimos anos resultou em um acúmulo de dados a respeito de diversos organismos. Atualmente esforços são realizados objetivando: extrair informações úteis desses dados, entender a relação entre os mesmos, entre outros.

É interessante notar os níveis de abstração utilizados para representar informações biológicas, funções e características das proteínas: uma proteína é composta por unidades denominadas aminoácidos, num total de vinte. Cada aminoácido corresponde a uma molécula especial que é mapeada para uma letra do alfabeto. Surge aí a representação primária da proteína como sendo a sequência linear de aminoácidos que a compõe.

De forma semelhante, a língua portuguesa usa um sistema de símbolos para comunicação, por meio de um alfabeto de vinte e seis letras. Assim como palavras na língua portuguesa possuem significado e utilidade, há informação útil contida na sequência de aminoácidos das proteínas. Esta metáfora surge como motivação para uma investigação a respeito da possível interseção entre soluções usadas em algoritmos de *stemming* e as usadas na Bioinformática, em especial em sequências de proteínas.

A busca pela interseção entre linguagens e biologia é um estudo gratificante e desafiador. No contexto desta tese, algumas questões surgem: é possível interpretar a sequência de uma proteína de forma análoga ao processo de *stemming*? Subsequências de uma proteína poderiam ser vistas como palavras (ou *stems*) desse genoma? Qual o significado desses *stems*? Uma solução para o processo de *stemming* pode ser útil para resolver algum problema da Bioinformática? A sequência de aminoácidos de uma proteína pode ser interpretada como sendo uma sentença de uma linguagem biológica?

Neste trabalho buscou-se explorar assuntos relacionados ao estudo de linguagens e Biologia Molecular, procurando interseção entre essas áreas, por meio da contextualização de soluções concebidas e aplicadas a problemas de *stemming* e ao problema de busca por *domínios* proteicos.

1.3 Bioinformática e portais *web*

A Bioinformática contempla um vasto campo de estudo que aplica técnicas da informática, envolvendo a análise e tratamento de informações oriundas de áreas de estudo da Biologia. Desenvolve também soluções em *software* para problemas relacionados às informações biológicas. No contexto deste trabalho, estão envolvidos dados referentes às características de macromoléculas, em especial, as proteínas.

Atualmente, com a facilidade de acesso à *web*, a obtenção de dados biológicos para uso em pesquisa é um processo disponível a qualquer interessado que tenha acesso à rede internet. Dados sobre proteínas, por exemplo, podem ser encontrados em diversos portais na rede, tais como PDB, Uniprot, Pfam (PUNTA *et al.*, 2012), entre outros. A maioria desses dados é de domínio público e de acesso livre. A seguir, breve caracterização de alguns portais que gerenciam informações sobre Biologia Molecular:

O PDB (*Protein Data Bank* http://www.rcsb.org/pdb/home/home.do), é um portal que disponibiliza dados sobre estruturas 3D de milhares de macromoléculas catalogadas. Tais informações, são geralmente obtidas por métodos de difração de raios x e ressonância magnética nuclear, e são enviadas por cientistas do mundo todo. Os dados são de domínio público e podem ser usados livremente. Já existe, inclusive, aplicativo móvel para acesso aos dados do PDB por meio de celulares.

O Uniprot (*Universal Protein Resource* http://www.uniprot.org/), é um portal que disponibiliza dados de sequências do Swiss-Prot e TrEMB, as quais foram manualmente anotadas. Normalmente, os portais apresentam ferramentas para acesso interativo, com funções especiais tais como anotação, alinhamento, ferramentas para

visualização de estruturas 3D, entre outras. Vale ressaltar que na maioria dos casos os portais fazem referência (*link*) para outros portais, visando integração de dados entre os mesmos.

Alguns portais disponibilizam, além do acesso interativo via *web*, a possibilidade de recuperar os arquivos que possuem informações sobre as estruturas biológicas de interesse, para uso local. Esta foi a opção mais adequada para o desenvolvimento desta tese, visto que o trabalho do pesquisador em tarefas específicas tais como a obtenção de amostra de dados, cálculo de informações estatísticas, além do próprio processo de recuperação dos dados biológicos para análise é mais livre e imediato.

Um dos desafios encontrados neste tipo de pesquisa está relacionado ao processo de gerência da vasta quantidade de dados disponível, o que requer estudo sistemático dos formatos de armazenamento dos mesmos bem como o desenvolvimento de procedimentos computacionais para recuperar as informações de interesse. Nos portais, existe uma variedade de formatos de arquivos que armazenam os dados biológicos, sendo que muitos encontram-se no formato texto, fasta, ou mesmo armazenados sob a forma de tabela para acesso por meio de Sistemas Gerenciadores de Banco de Dados relacionais (SGBDr). Este último refere-se ao formato de dados utilizado pelo Pfam.

O portal Pfam (http://pfam.sanger.ac.uk/), escolhido como fonte de dados das sequências utilizadas neste trabalho, contém alinhamentos de domínios de milhares de proteínas e perfis baseados nos modelos *Hidden Markov*, comumente denominados de *profile HMM*. A definição de limites de domínio, membros de família, bem como alinhamento é baseada na habilidade dos *profile HMMs* para identificar corretamente e alinhar os membros.

No Pfam, há três arquivos importantes para cada família de domínio de proteína: o primeiro, contém o *seed* alinhamento, que corresponde a um alinhamento múltiplo manualmente verificado de um conjunto representativo de sequências. O segundo, é um *profile HMM* construído do *seed* alinhamento, com o propósito de pesquisa e alinhamento para banco de dados. O último é o *full* alinhamento, o qual é gerado automaticamente do *seed* do *profile HMM*.

O portal Pfam disponibiliza diversas ferramentas para uso interativo. Por meio do portal é possível, por exemplo, buscar por sequências existentes no banco de dados, semelhantes a determinada sequência de interesse; visualizar informações das sequências pertencentes a determinada família de proteínas; buscar clans de famílias, ou

seja, grupos de famílias de proteínas que possuem semelhança; visualizar o domínio de determinada sequência de aminoácidos, entre outras formas de interagir.

Para que haja melhor ideia a respeito da vasta quantidade de dados, no Pfam versão 27, atualizado em março de 2013, há 14.831 famílias de proteínas. Nesta versão, os dados estão armazenados em 75 tabelas, e para cada uma há dois arquivos: arquivo de *script .SQL* o qual define os campos e tipos de dados da tabela; e o outro, que contém os dados. Os dados do Pfam usados neste trabalho, estão disponíveis na *web* para livre acesso, e podem ser obtidos no endereço: ftp://ftp.sanger.ac.uk/pub/databases/Pfam/releases/Pfam27.0/database_files/.

Uma vez que os dados usados nesta tese estão disponíveis no banco de dados do Pfam, o qual usa o SGDB MySQL, convém investigar os conceitos relativos ao modelo de dados usado por esta ferramenta para armazenar e gerenciar os dados.

1.4 Objetivos

São relatados a seguir o objetivo geral do trabalho e os objetivos específicos.

1.4.1 Objetivo Geral

O objetivo desta tese é testar a hipótese nula "A técnica de *stemming* não tem acurácia para identificar domínios de proteínas".

1.4.2 Objetivos Específicos

Adaptar uma solução de *stemming* para estimar domínios de sequências em uma amostra de proteínas, além do cálculo do poder de teste (erro β) para validação das conclusões.

1.5 Organização do texto

Este texto está dividido em seis partes, além desta introdução. O capítulo II abrange revisão bibliográfica sobre algoritmos de *stemming*, além de aspectos relacionados à Biologia Molecular - em especial as proteínas.

O capítulo III apresenta a metodologia do trabalho, envolvendo a concepção de um algoritmo para grupar *palavras*, aplicável à construção de algoritmos de *stemming*, além de um algoritmo de *stemming* adaptado para o problema de identificação de domínios em proteínas. No capítulo IV são relatados estudos de caso envolvendo as propostas apresentadas no capítulo III.

No capítulo V, são reportadas conclusões da pesquisa. Na penúltima parte, são mostradas as referências do trabalho. Por último, detalhados os materiais e métodos.

Capítulo II – Revisão Bibliográfica

2.1 Preâmbulo

Neste capítulo são apresentados conceitos básicos sobre o processo de *stemming* (sob uma perspectiva histórica), bem como os relativos fundamentais da Biologia Molecular.

2.2 Perspectiva Histórica sobre *Stemming*

Em 1968 foi publicado um algoritmo de *stemming* (LOVINS, 1968) para a língua inglesa, baseado numa série de princípios, a citar: remoção do maior sufixo da palavra; uso de lista de sufixos e regras, tamanho mínimo do *stem* e tratamento de exceções. Soluções dessa natureza são dependentes do idioma, e conhecidas por removedores de afixos.

Na década de 70, um método estatístico foi usado para o processo de *stemming* (HAFER & WEISS, 1974). Tal método, denominado variedade de sucessores, leva em consideração a posição de cada letra que compõe a palavra, além das letras antecessoras e sucessoras. Esta informação é utilizada para subdividir a palavra quando da geração do *stem*.

A pesquisa feita por Tars (TARS, 1976), destacou a importância do uso de *stemming* no processo de busca e recuperação da informação, visto que as palavras normalmente não perdem muito significado ao serem removidos os sufixos delas.

Em 1994 foi usada a técnica de n-grama (WILLIAN & JOHN, 1994) para a tarefa de classificação de texto, extensível ao processo de *stemming*. Um n-grama corresponde a uma sequencia de n caracteres de uma palavra. A ideia geral parte do princípio de que palavras que representam o mesmo conceito normalmente compartilham n-gramas. Por exemplo, as palavras carreta, carro, carrinho e carroça de forma aproximada apontam para o significado associado a carro, e compartilham o 4-grama *carr*.

Ainda em 1994, foi publicado um método de avaliação para algoritmos de *stemming* (PAICE, 1994). O método apresenta o cálculo de dois índices de erros: *overstemming index* (OI) e *understemming index* (UI), os quais são utilizados para comparar o desempenho de algoritmos a partir de uma amostra de palavras. O Método

de Paice tem sido extensivamente utilizado para avaliar *stemmers* desenvolvidos nos mais diversos idiomas e será analisado em detalhes neste trabalho.

O algoritmo clássico de extração de sufixos é conhecido por Algoritmo de Porter (PORTER, 1997). Tal algoritmo é constituído de cinco fases, nas quais são aplicadas regras às palavras para a remoção dos sufixos mais comuns. Com base em uma métrica específica relacionada ao número de vogais-consoantes presentes em uma palavra, o *stemmer* tenta evitar a remoção de letras caso o *stem* seja muito pequeno.

Existe um conjunto de palavras, conhecidas por *stop words*, as quais normalmente não constituem informação relevante para o contexto principal do texto e não são processadas por algoritmos de *stemming*. Para isso, os *stemmers* gerenciam uma lista de palavras (*stoplist*), identificando se a palavra é ou não uma *stop word*. Exemplos de *stop words* para a língua portuguesa, extraídos de (ALVARES, 2005): a, adeus, agora, aí, bem, boa, boas, certeza, cima, cinco, daquele, daqueles, dar, das, de.

Apesar de *stop words* em geral não contribuírem para o sentido principal do texto, em pesquisa utilizando textos no idioma inglês (RILOFF, 1995), foi verificado que algumas formas de substantivos, preposições e formas verbais são responsáveis por produzir resultados totalmente diferentes quando do uso dessa técnica no processo de classificação de texto.

Em 1996, foi publicado trabalho sobre um estudo comparativo envolvendo *stemmers* para a língua inglesa, baseados na técnica removedores de afixos (HULL & GREFENSTETTE, 1996). Foi analisado o uso desses algoritmos em um sistema de recuperação de informações. Além disso, os autores reportaram não ser uma boa ideia a remoção de prefixos por algoritmos de *stemming*.

Em pesquisa publicada em 1998 (FULLER & ZOBEL, 1998), foram analisadas diferentes vantagens e desvantagens de vários *stemmers* desenvolvidos para a língua inglesa. O estudo de tais algoritmos foi realizado em função do uso dos mesmos em um sistema de recuperação de informações. Ainda em 1998, surgiu uma abordagem estatística, independente do idioma, tendo sido testada em textos das línguas inglesa e espanhola (XU & CROFT, 1998). O processo de *stemming*, em publicação na área de recuperação de informações (ZIVIANI & RIBEIRO-NETO, 1999) foi relatado como parte da atividade de pré-processamento de textos.

Em 2000, um algoritmo de *stemming* foi utilizado como parte do processo de integração de dados de fontes heterogêneas, para gerar termos de documentos

(WILLIAM, 2000). O trabalho de Kantrowitz e colaboradores (KANTROWITZ *et al.*, 2000) avaliou de forma empírica *stemmers* desenvolvidos para a língua inglesa.

A pesquisa desenvolvida por Goldsmith e equipe apresentou um modelo de identificação de sufixos a partir de uma amostra de palavras, ocorrendo de forma independente do idioma (GOLDSMITH, HIGGINS & SOGLASNOVA 2000). Além disso, apresentou breve histórico sobre *stemmers* removedores de afixos, bem como soluções estatísticas, não dependentes da morfologia do idioma alvo.

No trabalho de Frakes (FRAKES & FOX, 2003), foram avaliados quarto *stemmers* (Lovins, Paice, Porter e S-removal) para a língua inglesa. Além disso, na pesquisa foram propostas métricas para avaliação de *stemmers*.

Um algoritmo de *stemming,* chamado RSLP – Removedor de Sufixos da Língua Portuguesa, desenvolvido especialmente para a língua portuguesa foi apresentado por Orengo (ORENGO & HUYCK 2001). No RSLP, o processo de *stemming* ocorre em oito passos: redução de plural, feminino, aumentativo, advérbio, substantivo, verbo, vogais e acentos. Foi realizado experimento com uma amostra de 1000 palavras, envolvendo uma versão do algoritmo de Porter para o Português e o algoritmo proposto.

A pesquisa realizada por Chaves (CHAVES, 2003), avaliou dois algoritmos de *stemming* desenvolvidos para a língua portuguesa (um de autoria do pesquisador Marco Antonio Insaurriaga Gonzalez; o outro, RSLP), por meio da realização de um experimento com uma amostra de 500 palavras.

Outro algoritmo, desenvolvido para a língua portuguesa, foi encontrado no trabalho de Alvares (ALVARES, 2005). Denominado STEMBR, o *stemmer* utiliza regras para extração de sufixos, e para a construção de tais regras baseia-se num estudo estatístico realizado em palavras da língua portuguesa, grupadas pela mesma letra final.

Ainda no contexto da língua portuguesa, o trabalho de (ALVARES & GARCIA 2005), avaliou dois *stemmers* (RSLP e STEMBR), por meio da realização de um experimento com uma amostra de 5000 palavras. Ainda em 2005, foi publicada pesquisa que realizou uma generalização do Método de Paice (DE MADARIAGA, CASTILLO & HILERA 2005).

Os primeiros algoritmos de *stemming* foram desenvolvidos para a língua inglesa (PORTER, 1997) (SRINIVASAN & THAMBIDURAI 2006). Com o passar do tempo, surgiram *stemmers* para diversos idiomas, tais como: alemão, árabe, espanhol, português, entre outros. As primeiras propostas foram construídas de forma atrelada a conhecimentos morfológicos do idioma utilizado. No entanto, existem pesquisas que

adotam outra abordagem, evitando ao máximo vínculo com tais estruturas. Algumas são relatadas a seguir:

Na pesquisa desenvolvida por Bacchin e colaboradores foi apresentado um *stemmer* estatístico, baseado em análise de *links* (BACCHIN, FERRO & MELUCCI 2002). De posse de um conjunto de palavras, o algoritmo obtém, para cada uma, as possíveis subcadeias candidatas a *stem*. As melhores candidatas são as que possuem frequência alta, e formam palavras com os sufixos que possuem frequência alta.

O relatório técnico de Silva e equipe (SILVA & OLIVEIRA 2003) reportou o uso de léxico para construção de *stemmers*, com proposta baseada na retirada de afixos e consulta a tabela. O processo de *stemming*, visto como suavização estatística, foi apresentado em (ALLAN & KUMARAN 2003).

O trabalho de Mayfield e equipe (MAYFIELD & MCNAMEE 2003), reportou o uso de n-grama na construção de *stemmer*. A vantagem é que em geral determinado n-grama contém uma subcadeia de caracteres presente na palavra e nas diversas variações da mesma. Por exemplo, as palavras *computador, computadorizado, computar* e *computacional* compartilham o 6-grama *comput*.

Stemmers estatísticos gerados a partir do uso de Cadeias de Markov Escondidas foram reportados por Melucci e colaboradores (MELUCCI & ORIO 2003). Baseada numa lista de palavras, requerida como conjunto de treinamento, a cadeia calcula o *stem* mais provável para uma palavra arbitrária. Este *stemmer* baseia-se no conceito de HMM, que são autômatos finitos onde as transições entre os estados são calculadas por funções de probabilidade. Em cada transição, o novo estado emite um símbolo com uma determinada probabilidade.

Hammarstron apresentou solução de *stemming*, a qual ocorre em duas fases: na primeira, uma lista de afixos é retirada do texto da linguagem em questão; na segunda, pares de palavras são comparados com a lista obtida na primeira fase, objetivando identificar se possuem o mesmo *stem* (HAMMARSTRON, 2006).

Uma abordagem, baseada em clusterização, para obter classes de equivalência de palavras com mesma raiz e suas variantes morfológicas, foi apresentada por Majumder e colaboradores. Para a clusterização das palavras em grupos homogêneos, um conjunto composto por quatro distâncias de similaridade entre palavras foi proposto e utilizado (MAJUMDER *et al.,* 2007).

Na pesquisa realizada em (METZLER, DUMAIS, MEEK 2007), diversas técnicas de similaridade entre segmentos curtos de textos, incluindo *stemming*, foram

avaliadas. Ainda em 2007, foram realizados experimentos envolvendo diversas soluções de *stemming* em textos da língua portuguesa (ORENGO, BURIOL & COELHO 2007). Já o uso de *stemming*, no contexto da recuperação de informação em mecanismos de buscas *web*, foi investigado por Peng e colaboradores (PENG *et al.*, 2007). Outra solução, foi reportada por Mcnamee e equipe (MCNAMEE, NIVHOLAS & MAYFIELD 2008). O algoritmo, chamado *Morfessor*, requer como entrada uma lista de palavras do idioma em questão. Como saída, fornece a palavra segmentada.

Na pesquisa realizada por Sharma (SHARMA, 2012), são reportados os principais trabalhos publicados no processo de concepção de algoritmos de *stemming*. São caracterizados *stemmers* dependentes da morfologia do idioma em questão (baseados em regras), bem como soluções estatísticas, sem vínculo com os prefixos e sufixos do idioma.

Recentemente, no trabalho desenvolvido por Xavier e colaboradores (XAVIER, SILVA & GOMES, 2013), foi realizada análise comparativa de algoritmos de *stemming*, bem como relatada a importância dessas soluções no processo de mineração de textos.

2.3 Avaliando Algoritmos de *Stemming*

Em se tratando de avaliação de algoritmos de *stemming*, o método mais difundido para realizar tal tarefa é o de Paice, conforme a seguir.

2.3.1 Método de Paice

Em 1994, Paice publicou um método para avaliação de algoritmos de *stemming* (PAICE, 1994), em que quatro medidas foram introduzidas: *Overstemming Index* (OI), *Understemming Index* (UI), *Stemming Weight* (SW) e *Error Rate Relative to Truncation* (ERRT). As medidas mais comumente utilizadas são as duas primeiras.

O método requer uma amostra de palavras, sem repetição, dividida em grupos conceituais onde as palavras de cada grupo são semântica e morfologicamente relacionadas. Assim, desconsiderando homógrafos, são definidas duas classes de relacionamentos:

• **Classe 0**: Palavras diferentes em forma, mas semanticamente equivalentes, e;

• **Classe 1**: Palavras diferentes em forma e semanticamente distintas.

Um bom algoritmo de *stemming* é aquele que na medida do possível gera apenas um *stem* para palavras pertencentes à **Classe 0**, evitando gerar o mesmo *stem* para palavras pertencentes à **Classe 1**. Para cada grupo conceitual, dois totais são calculados:

• *Desired Merge Total (*DMT), que representa o número possível de pares diferentes formados por palavras do mesmo grupo, obtido por:

$$\mathrm{DMT_g} = 0{,}5n_g(n_g - 1), \quad (\mathbf{1})$$

onde n_g representa o número de palavras do grupo. Tal valor corresponde ao número de pares de palavras de mesmo *stem* que queremos ter, pois é desejável que todas as palavras de mesmo grupo tenham o mesmo *stem*. Para grupos que possuem apenas uma palavra, o valor DMT é zero.

• *Desired Non-merge Total* (DNT), que representa o número de pares formados por uma palavra membro com outra não-membro do grupo, obtido por:

$$\mathrm{DNT_g} = 0{,}5n_g(W - n_g), \quad (\mathbf{2})$$

onde *W* representa a quantidade de palavras da amostra. Tal valor corresponde ao número de pares de palavras que não desejamos que tenham o mesmo *stem*.

Somando-se os valores DMT de cada grupo, é obtido o GDMT (*Global Desired Merge Total*). Da mesma forma, somando-se os valores DNT de cada grupo, é obtido o GDNT (*Global Desired Non-merge Total*).

Após a execução do algoritmo, um *stem* é gerado para cada palavra. Entretanto, há possibilidade de existirem *stems* diferentes em um mesmo grupo conceitual. Esta inabilidade é quantificada pelo parâmetro UMT (*Unachieved Merge Total*), obtido por:

$$UMT_g = 0{,}5 \sum_{i=1}^{s} u_i(n_g - u_i), \quad (3)$$

onde **s** é o número de *stems* distintos no grupo e u_i é o número de instâncias de cada *stem* no grupo. Novamente, somando-se os valores UMT de cada grupo, é obtido o GUMT (*Global Unachieved Merge Total*). O UI é calculado pela divisão GUMT/GDMT.

Podem existir casos em que o mesmo *stem* tenha sido gerado para palavras pertencentes a dois ou mais grupos conceituais diferentes, resultando em *overstemming*. Para contabilizar tais erros, a amostra é reorganizada em grupos que compartilham o mesmo *stem*, e calculado o WMT (*Wrongly Merged Total*), obtido por:

$$WMT_s = 0{,}5 \sum_{i=1}^{t} v_i(n_s - v_i), \quad (4)$$

onde t é o número de grupos conceituais que compartilham o mesmo *stem*, n_s é o número de instâncias do *stem* e v_i o número de *stems* de cada grupo t. Somando-se os valores WMT, é obtido o GWMT (*Global Wrongly Merged Total*). O valor de OI é calculado por GWMT/GDNT. Finalmente, o valor SW é obtido pela divisão OI/UI.

O valor SW representa o "peso" do *stemmer*. Um *stemmer* é dito "pesado", quando possui OI alto e UI baixo. Já o considerado "leve", possui OI baixo e UI alto. Convém observar que SW alto ou baixo não tem relação com a qualidade do *stemmer*, ou seja, não implica que ele seja melhor ou pior.

O cálculo do ERRT parte da ideia que os valores dos pontos (UI,OI) de uma série de algoritmos de truncamento de palavras de diferentes tamanhos determinam uma linha que pode ser utilizada para medir a qualidade de um *stemmer*. Idealmente as coordenadas (UI, OI) do *stemmer* deveriam estar abaixo dessa linha de truncamento. Quanto mais distante (abaixo) dessa linha de truncamento, melhor é o *stemmer*. Por outro lado, quanto mais afastado, pior é o *stemmer*. O ERRT é obtido ao ser estendida uma linha da origem, passando pelo ponto P(UI,OI), até que ela interseccione a linha de truncamento no ponto T. O ERRT é obtido por:

$$ERRT=tamanho(OP)/tamanho(OT) \quad (5)$$

A Figura 2.1 a seguir, representa o cálculo de ERRT.

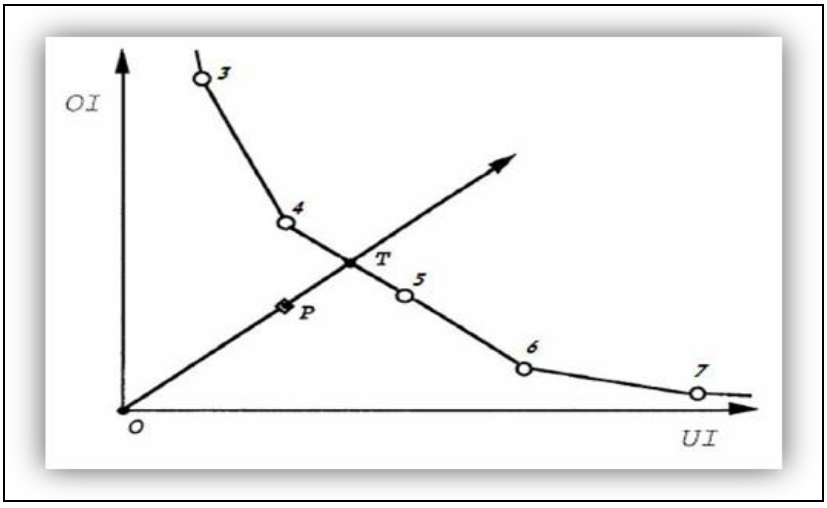

Figura 2.1: Cálculo de ERRT (PAICE, 1994)

Exemplos de trabalhos que usaram o método de Paice para auxiliar na avaliação de algoritmos de *stemming* podem ser encontrados em (ORENGO & HUYCK 2001) e (ALVARES, GARCIA & FERRAZ 2005).

2.4 Conceitos da Biologia Molecular

O ácido desoxirribonucleico (DNA), presente nas células, é formado por uma hélice dupla composta por um esqueleto de açúcares e fosfato e por pares de moléculas chamadas bases (Adenina, Timina, Guanina e Citosina). As duas metades da hélice são complementares, visto que cada um dos tipos de bases pode realizar emparelhamento apenas com a base do tipo complementar. A figura 2.2 a seguir representa o esquema da molécula de DNA, extraído do trabalho de Watson e Crick (WATSON & CRICK, 1953).

Figura 2.2 Molécula de DNA (WATSON & CRICK, 1953)

Na sequencia de bases do DNA, diferentes segmentos representam unidades funcionais independentes, denominadas genes. Os genes possuem toda a informação necessária para o processo de síntese de proteínas, que são macromoléculas essenciais para os seres vivos.

No processo de produção de uma proteína, uma das cadeias da dupla hélice de DNA serve como molde para síntese de uma sequencia de base complementar, chamada ácido ribonucleico mensageiro (RNAm), em um processo conhecido por transcrição. Cada trinca ordenada de bases do RNAm, chamada códon, codifica uma molécula denominada aminoácido, havendo total de 20, sendo cada um deles representado por uma letra: {*A, R, D, N, C, E, Q, G, H, I, L, K, M, F, P, S, T, W, Y, V*} (SETUBAL & MEIDANIS, 1997). Aminoácidos se ligam uns aos outros, formando cadeias polipeptídicas. As proteínas são formadas por uma ou mais cadeias polipeptídicas.

Os passos ora descritos representam o dogma central da Biologia Molecular, no que diz respeito ao fluxo de informação da célula, em que a molécula de DNA é usada como molde para a construção da molécula de RNA, a qual será usada para a síntese de proteínas. Os 20 aminoácidos estão representados na tabela 2.1 a seguir:

Tabela 2.1 Aminoácidos que constituem as proteínas

Código	**Letra**	**Abreviação**	**Nome**
1	A	Ala	Alanina
2	C	Cys	Cisteína
3	D	Asp	Ácido aspártico
4	E	Glu	Ácido glutâmico
5	F	Phe	Fenilalanina
6	G	Gly	Glicina
7	H	His	Histidina
8	I	Ile	Isoleucina
9	K	Lys	Lisina
10	L	Leu	Leucina
11	M	Met	Metionina
12	N	Asn	Asparagina
13	P	Pro	Prolina
14	Q	Gln	Glutamina
15	R	Arg	Arginina
16	S	Ser	Serina
17	T	Thr	Treonina
18	V	Val	Valina
19	W	Trp	Triptofano
20	Y	Tyr	Tirosina

Proteínas desempenham funções específicas dentro de um organismo. Elas podem ter um caráter estrutural (como a queratina presente nos cabelos) ou podem estar ligadas a algumas atividades, por exemplo, anticorpos, hormônios e enzimas, entre outras.

Um aminoácido é composto por um carbono central (C_α), um hidrogênio (H), um grupo amino (H_2N), um grupo carboxila ($COOH$) e uma cadeia lateral R, conforme a Figura 3.2.

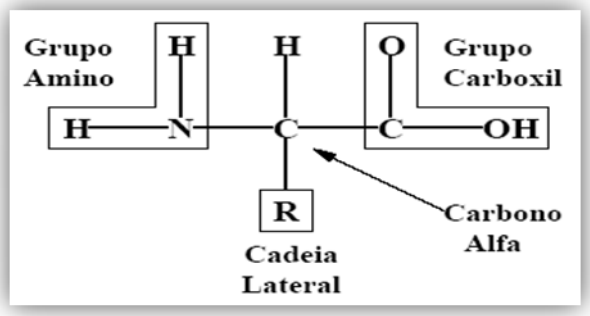

Figura 2.3 Estrutura química geral dos aminoácidos (ALBERTS *et al*, 2011)

A sequencia de aminoácidos representa a estrutura primária das proteínas, e define sua estrutura tridimensional. Formas tridimensionais estão diretamente associadas à função da proteína. O fato de existirem 20 diferentes aminoácidos propicia grande variedade de formas irregulares, que determinam a sua ligação com outras moléculas.

Ao se estudar a função de uma proteína, é importante conhecer quais as suas possíveis conformações. As proteínas podem ser consideradas em quatro níveis de arquitetura: estrutura (a) primária, (b) estrutura secundária, (c) estrutura terciária e (d) estrutura quaternária, conforme figura 2.4.

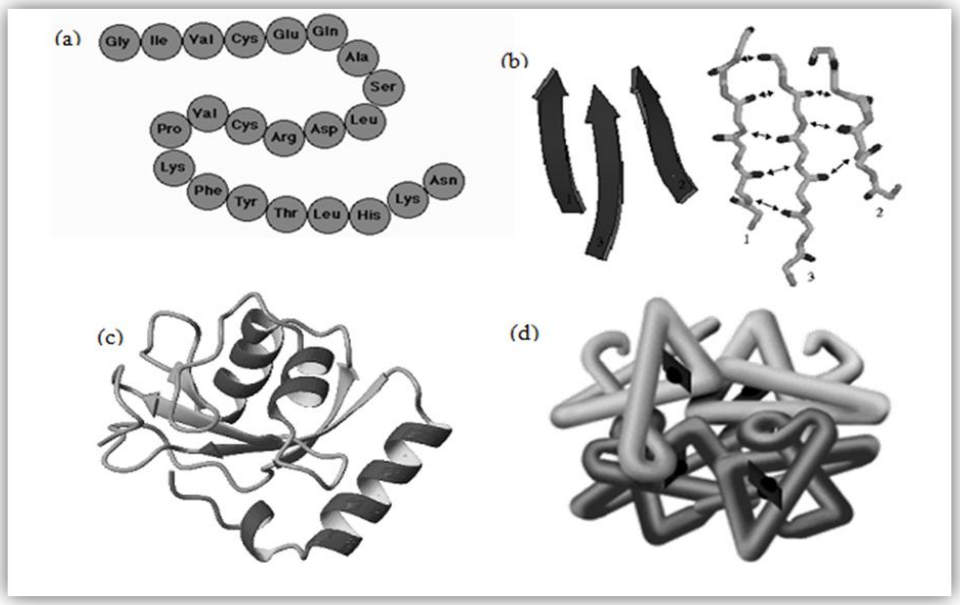

Figura 2.4 Níveis de representação de uma proteína (ALBERTS *et al*, 2011)

Na estrutura primária, as proteínas são representadas por aminoácidos. A diferença entre eles é determinada por sua cadeia lateral ou grupo *R*. Os grupos *R*

influenciam, por exemplo, na solubilidade do aminoácido em água e possuem diferentes estruturas. Ligações peptídicas unem aminoácidos durante a síntese de proteínas. Neste processo, ocorre aparecimento de uma molécula de água formada por um hidrogênio (*H*) do grupo amino com a hidroxila (*OH*) do grupo carboxila. Por isso, os aminoácidos são muitas vezes referenciados como resíduos. O resíduo localizado na extremidade que exibe o grupo amino é denominado amino-terminal ou *N*-terminal. Na outra extremidade, o resíduo que exibe o grupo carboxila é denominado carboxila-terminal ou *C*-terminal.

A estrutura primária é convencionalmente lida da extremidade *N*-terminal para a *C*-terminal, conforme Figura 2.5. O arranjo recorrente de resíduos no espaço é denominado estrutura secundária, conforme Figura 2.6.

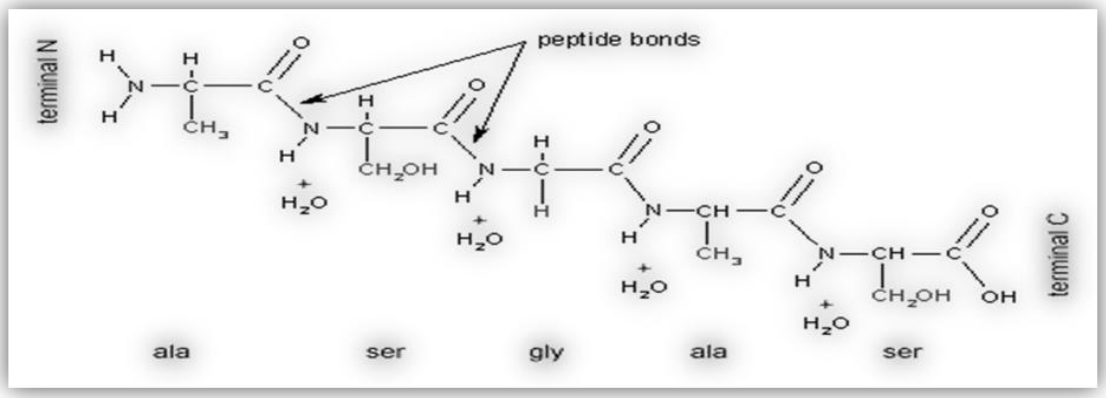

Figura 2.5: Estrutura primária de uma proteína (ALBERTS *et al*, 2011)

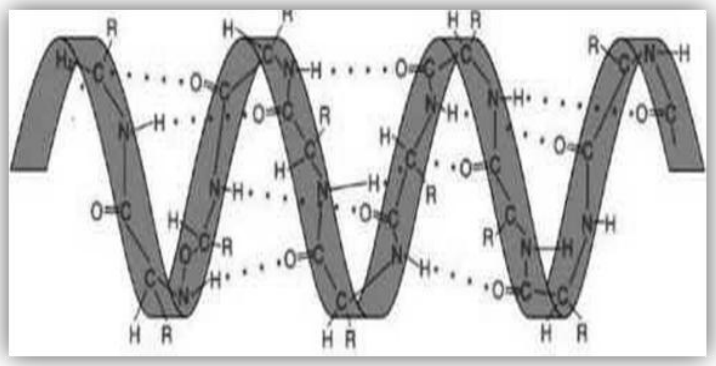

Figura 2.6 Estrutura secundária de uma proteína (ALBERTS *et al*, 2011)

Os elementos regulares encontrados na estrutura secundária são distribuídos em três classes: α-hélices, folhas-β e *coils*.

As α-hélices (vide figura 2.7) são estabilizadas por pontes de hidrogênio paralelas ao seu eixo que ocorrem no interior do *backbone* de uma única cadeia

polipeptídica. Contando-se a partir da extremidade *N*-terminal, o grupo *C=O* de cada resíduo de aminoácido está ligado por pontes de hidrogênio ao grupo *N-H* na sequencia linear mantida por ligações covalentes.

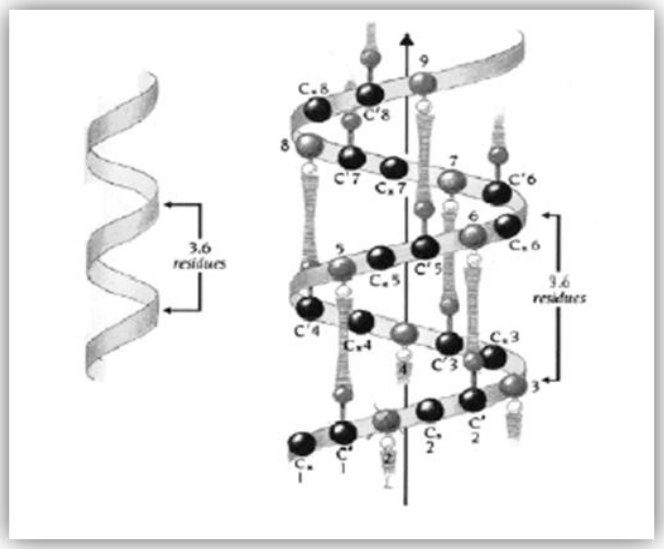

Figura 2.7 Representações esquemáticas de uma α-Hélice (ALBERTS *et al*, 2011)

Folhas-*β* são formadas pela combinação de várias regiões não contíguas, denominadas *β-strands*. Esses elementos são formados por 5 a 10 resíduos adjacentes e são ligados a outros *β-strands* através de pontes de hidrogênio (vide figura 2.8).

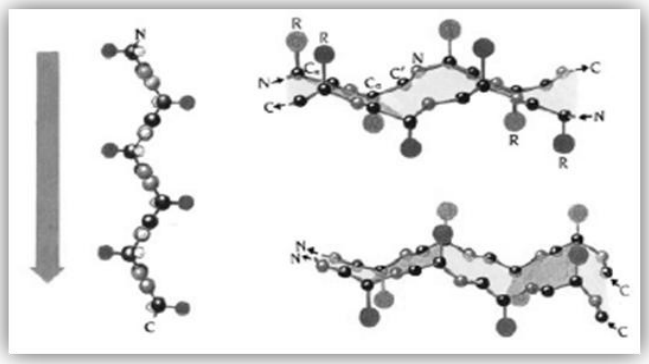

Figura 2.8 β-Folhas e as ligações entre os β-strands (ALBERTS *et al*, 2011)

A classe *coil* abrange os demais arranjos recorrentes que não foram incluídos nas duas primeiras, tais como hélices que não são energeticamente favoráveis e segmentos irregulares como os *loops*, que podem assumir formas e tamanhos diversos. *Coils* são estruturas irregulares com a não repetição dos ângulos de torção do *backbone* e, em geral, têm uma ligação de hidrogênio.

Basicamente, a estrutura terciária de uma proteína consiste da conformação tridimensional dos elementos de estrutura secundária em uma única cadeia polipeptídica. No entanto, é possível identificar subestruturas recorrentes. Existem alguns níveis intermediários de estruturas: as estruturas supersecundárias e os domínios. As primeiras, denominadas *motifs* (motivos), são conjuntos estáveis de elementos de estrutura secundária, os quais formam arranjos particulares através de interações entre as cadeias laterais.

Motifs podem ocorrer várias vezes em uma mesma proteína ou em proteínas diferentes. Além disso, podem estar ou não associados a uma função. Já os domínios são regiões compactas que podem compreender de 40 a 400 aminoácidos formando uma unidade estrutural distinta na região conservada da proteína (o *core*). Domínios estão associados a uma função e, alguns arranjos são energeticamente mais favoráveis do que outros.

Diversas proteínas podem ser grupadas em famílias de proteínas. Cada membro de uma família possui sequência de aminoácidos e conformação tridimensional semelhantes aos outros membros da família.

2.5 Proteínas e bancos de dados

Em termos de representação computacional, as sequências primárias de aminoácidos são tratadas como *strings* de caracteres. No entanto, há diversas outras informações de interesse associadas, tais como dados sobre famílias, tamanho da sequência, quantidade de sequências por família, nome da família, autor, características de hidrofobicidade, estruturas e funções de macromoléculas, informações sobre clans, entre outras.

Uma vez que as relações entre sequências e suas características são estabelecidas, o mecanismo computacional atual mais eficiente para guardar e gerir os dados e seus relacionamentos se dá por meio do uso de um Sistema Gerenciador de Banco de Dados (SGBD) relacional.

O Pfam, por exemplo, armazena os dados no SGBD relacional MySQL, que é um *software* livre, o qual pode ser baixado e usado a partir do site http://www.mysql.com/. Para isto, o usuário deve realizar cadastro no portal, e escolher a versão adequada para seu sistema operacional.

Capítulo III – Metodologia do Trabalho

3.1 Preâmbulo

Neste capítulo, são apresentadas duas propostas: concepção de um modelo para o processo de *stemming*, sem vínculo com prévios conhecimentos da estrutura morfológica do idioma em questão. Além disso, é apresentada a adaptação de um *stemmer* ao problema de identificação de domínios de proteínas.

3.2 Modelo Abr para *Stemming*

As soluções de *stemming* ora apresentadas são dependentes da estrutura morfológica do idioma em questão, normalmente vinculadas ao conhecimento de prefixos e sufixos, bem como a busca por um processo sistemático para realização do *melhor* corte.

Uma contribuição deste trabalho é a concepção de um modelo para obtenção de *stems*, sem vínculo com a estrutura morfológica do idioma. O modelo, denominado modelo Abr, será instanciado na língua portuguesa, e detalhado conforme a seguir:

É estabelecido que a partir de um conjunto de palavras de determinado idioma *i*, dividido em *g* grupos conceituais (palavras relacionadas morfologicamente), as palavras de cada grupo compartilham diversos *stems*.

Para ilustrar, a **Figura 3.1 (a)** a seguir mostra três grupos conceituais (infernizar, informar e ingressar (*g=3*)) do idioma português. Nota-se que, independente dos prefixos e sufixos, há um padrão compartilhado por cada palavra pertencente a determinado grupo (**Figura 3.1 (b))**. Dada tal constatação, foi estabelecida hipótese de que esse padrão, uma vez extraído, seja um bom *stem* que possa representar as palavras de cada grupo. Para extraí-lo encontraremos a maior subcadeia comum em posição contígua entre as palavras de cada grupo.

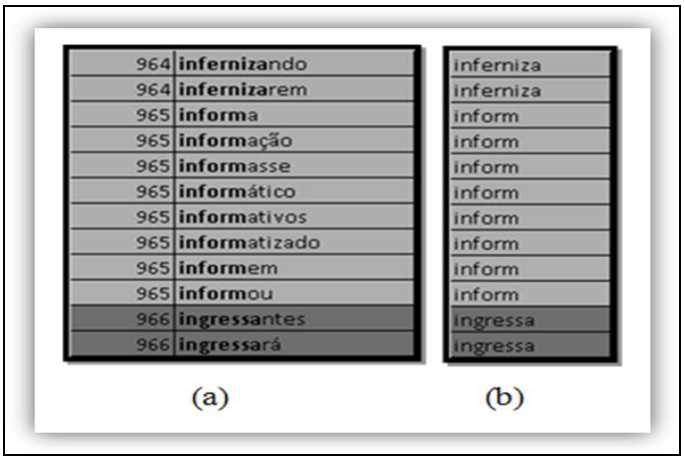

Figura 3.1 Exemplo de três grupos conceituais

A **Figura 3.2** a seguir exibe um fluxograma para extração de *stems* no contexto da abordagem apresentada:

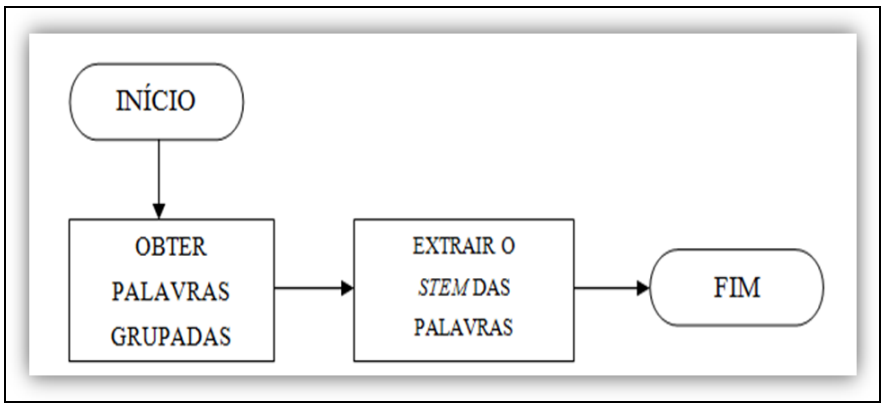

Figura 3.2 Fluxograma para extração do *stem*

A partir deste cenário, as seguintes hipóteses de trabalho foram concebidas:

- Hipótese 01: *a tarefa de divisão das palavras em grupos conceituais (obter palavras grupadas) pode ser realizada de forma automática, a partir da aplicação de um processo de clusterização de léxico, baseado na definição de uma função de distância entre cada par de palavras do conjunto.*
- Hipótese 02: *Um stem, dentre os n possíveis, que possa representar cada grupo g, corresponde à maior subcadeia comum contígua existente entre as palavras do grupo(extrair o stem das palavras).*

O Modelo proposto, denominado Modelo Abr, é ilustrado na Figura 3.3 a seguir:

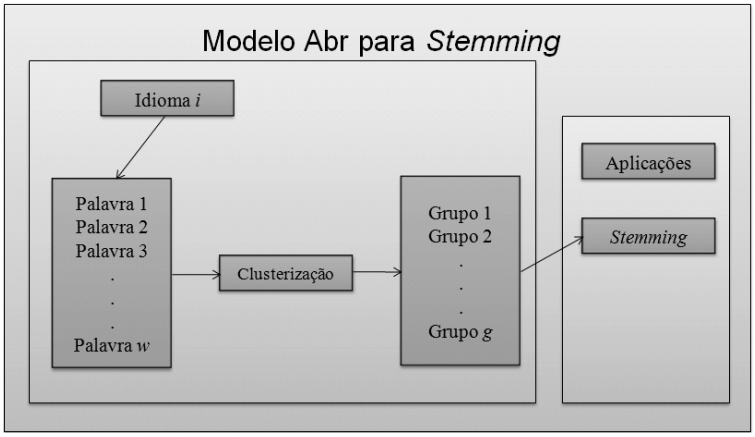

Figura 3.3 Geração de grupos e aplicações no processo de *stemming*

Os seguintes passos resumem os componentes do modelo:

- Escolha do idioma: aqui é definido qual idioma será usado. No contexto da tese, o idioma português foi escolhido para teste.
- Escolha das palavras: escolhido o idioma, obtêm-se as palavras que serão usadas no processo de clusterização, objetivando a formação dos grupos. São retiradas palavras que normalmente não tenham grande importância para o processo (*stop words*).
- Processo de Clusterização: realizados os passos anteriores, deve-se estabelecer um critério para comparar pares de palavras, quantificando o quão são semelhantes, pois o processo de clusterização requer a existência de algum tipo de similaridade entre os dados. Neste caso, o modelo prevê o uso de funções de distância entre *strings*.
- Aplicações: com os grupos criados, parte-se da hipótese de extrair o *stem* de cada grupo conceitual.

A seguir, serão detalhados os principais passos para a validação do modelo.

3.2.1 Similaridade entre *strings*

Funções de distância entre um par de *strings* X e Y correspondem a um número real *r*, de forma que, caso *r* seja pequeno, X e Y são similares. Uma das mais conhecidas é a Distância de Levenstein (LEVENSTEIN, 1966) ou Distância de Edição (DEdit), que corresponde ao número mínimo de operações de edição (inserção, remoção, ou substituição) necessário para transformar uma *string* em outra. Para ilustrar, DEdit <mergulhei, mergulhador> = 4.

Recentemente, Majumder e colaboradores (MAJUMDER *et al* 2007), definiram quatro funções (D_1, D_2, D_3, D_4), descritas como segue: a partir de duas *strings* $X = x_0x_1 \ldots x_n$ e $Y = y_0y_1 \ldots y_{n'}$ é definida uma função lógica p_i, usada para penalidade:

$$p_i = 0 \text{ se } x_i = y_i \text{ com } 0 \leq i \leq \min(n,n'), 1 \text{ caso contrário.}$$

A p_i vale 1 se na *i-ésima* posição entre as *strings* X e Y as letras são diferentes. Caso X e Y tenham tamanhos diferentes, a menor *string* é completada com nulos para que fiquem com mesmo comprimento. Seja *m* a posição em que $x_i \neq y_i$. Admitindo que o comprimento das *strings* seja $n + 1$. As funções são definidas como segue:

$$D_1(X,Y) = \sum_{i=0}^{n} \frac{1}{2^i} p_i$$

p_i=1 se houver caracteres diferentes na mesma posição, ou seja, $x_i \neq y_i$

$$D_2(X,Y) = \frac{1}{m} \times \sum_{i=m}^{n} \frac{1}{2^{i-m}} \quad se\ m > 0, \infty\ para\ qualquer\ outro\ resultado$$

$$D_3(X,Y) = \frac{n-m+1}{m} \times \sum_{i=m}^{n} \frac{1}{2^{i-m}} \quad se\ m > 0, \infty\ para\ qualquer\ outro\ resultado$$

$$D_4(X,Y) = \frac{n-m+1}{n+1} \times \sum_{i=m}^{n} \frac{1}{2^{i-m}}$$

Convém observar que, excluindo D_1 as outras três distâncias não levam em consideração acertos em qualquer posição anterior à ocorrência do primeiro erro. A intuição por trás das definições de (D_1, D_2, D_3 e D_4) é premiar grandes (subcadeias de) prefixos e penalizar os erros tão logo ocorram no início das *strings*.

Aqui, tem-se o resultado de (D_1, D_2, D_3 e D_4) aplicado no par <mergulhei, mergulhador>: $D_1 = 0{,}01465$; $D_2 = 0{,}26786$; $D_3 = 1{,}07143$; $D_4 = 0{,}68182$.

Na Figura 3.4 a seguir, formada pelos caracteres do par <mergulhei, mergulhador>, a região circulada ilustra a primeira ocorrência de letra diferente entre as palavras.

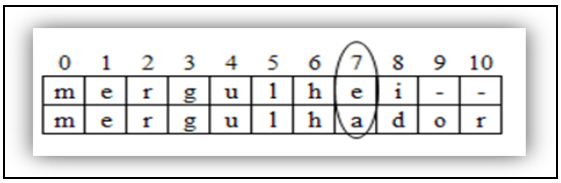

Figura 3.4 Par <mergulhei, mergulhador> primeira letra diferente entre palavras

Uma função foi definida, chamada DAbr, (ALVARES & MONDAINI 2010), apresentada conforme a seguir:

Dadas duas *strings* X e Y, se o problema a ser resolvido fosse apenas verificar se X e Y são iguais ou diferentes, poderíamos conceber uma função de distância binária tal que caso X = Y (palavras iguais), o resultado seria 0, e 1 caso contrário. Note que para tal função o valor 1 é válido tanto para o par **<mergulhei, mergulhador>**, quanto para **<mergulhei, estudei>**. No entanto, existe a necessidade de verificar o quão parecidos são os pares de palavras. No exemplo apresentado, é necessário capturar a informação de que o primeiro par é mais similar que o segundo.

Assim, surge a ideia de construir uma função, dividida em três partes, cujo domínio seja um real r¸ variando entre 0 (palavras iguais) e 1 (palavras totalmente diferentes em termos gráficos). A hipótese consiste em identificar e beneficiar uma região R, de tamanho LEN(R), comum entre as palavras, excluindo os casos em que as palavras são iguais. A região R corresponde à maior subcadeia comum em posição contígua entre duas *strings* (Figura 3.5).

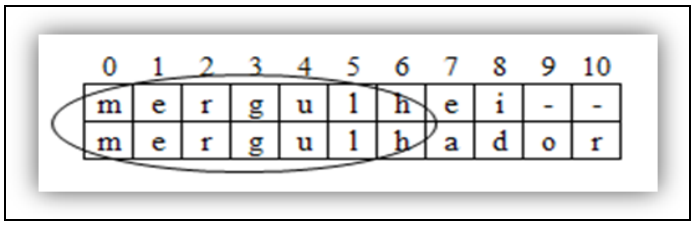

Figura 3.5 Par <mergulhei, mergulhador> e a maior subcadeia comum

A função DAbr(X,Y) é definida por:

$$\text{DAbr}(X,Y) = \begin{cases} 0, \text{ se } X=Y \text{ (palavras iguais) (I);} \\ 1, \text{ se } X \neq Y \text{ (letras diferentes em cada posição) (II). Caso contrário} \\ 1 - \sum_{i=1}^{\text{LEN(R)}} \left(1/2^{i}\right) \text{ (III)} \end{cases}$$

Para o exemplo DAbr<*mergulhei, mergulhador*>, a região R é a subcadeia *mergulh*, cujo tamanho LEN(R) é igual a sete, neste caso caindo no caso geral (III). Assim, tem-se que DAbr<*mergulhei, mergulhador*>=0,00781.

Com o objetivo de expressar o comportamento da função DAbr, foi construído o gráfico da Figura 3.6, contendo os valores possíveis para a função, considerando o parâmetro LEN(R) variando entre 1 e 7 (III) aplicado a um par de palavras cujo maior tamanho é 10(II).

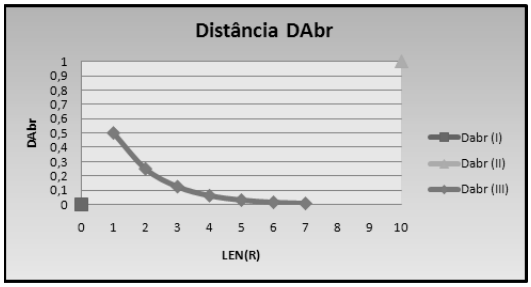

Figura 3.6 Valores possíveis para a função DAbr(X,Y), com max(LEN(R)) = 7.

Nota-se que o resultado de DAbr no caso (III) decresce a partir de 0,5; 0,25; 0,125; 0,0625, tendendo a ficar próximo de zero de acordo com o aumento da região LEN(R). No restante desse trabalho, DAbr será referenciada por D_5.

3.2.2 Análise de Precisão das Distâncias (D_1, D_2, D_3, D_4 e D_5)

Nesta seção é apresentada uma análise de precisão das distâncias (D_1, D_2, D_3, D_4 e D_5). Convém esclarecer os critérios utilizados para avaliar as medidas. Dada a própria definição de função de distância, quanto mais parecidas as palavras, menor a distância entre elas. Sendo diferentes, a distância tende a ser grande. Para isso, foram selecionados do trabalho de Alvares e colaboradores dois tipos de pares de palavras: as parecidas graficamente e as totalmente distintas, num total de 500 palavras (ALVARES, GARCIA & FERRAZ 2005).

No experimento, foram processados 124.750 pares. Visando facilitar a análise, foi elaborada a **Tabela 3.1**, com exemplos de pares de palavras, retirados da amostra.

Tabela 3.1 Resultados envolvendo pares retirados aleatoriamente da amostra

X	**Y**	**D_1**	**D_2**	**D_3**	**D_4**	**D_5**
abacaxi	apostavam	0,99609	1,99219	15,93750	1,77083	1.0
renderá	sinalize	1,74219	∞	∞	1,99219	1.0
renderá	sanguíneas	1,74805	∞	∞	1,99805	1.0

saturada	saturadas	0,00391	0,12500	0,12500	0,11111	0,00391
caracterizado	caracterizadores	0,00021	0,13462	0,40385	0,32813	0,00012

A partir da Tabela 3.1, nota-se que os valores das medidas estão de acordo com a definição de funções de distância, uma vez que aumentam quando os pares de palavras apresentam diferentes letras na mesma posição (vide os três primeiros pares). Da mesma forma, diminuem quando os pares de palavras apresentam letras iguais na mesma posição (vide os dois últimos pares). Optou-se também por analisar pares de palavras correspondentes aos valores mínimo e máximo das funções, conforme expresso na Tabela 3.2 a seguir:

Tabela 3.2 Valores mínimos e máximos das distâncias aplicadas na amostra

	D_1	**D_2**	**D_3**	**D_4**	**D_5**
Mínimo	0,00004	0,10000	0,06250	0,00714	0,00003
Máximo	1,99996	∞	∞	2.0	1.0

A Tabela 3.3 a seguir exibe exemplos para cada um dos valores mínimo e máximo encontrados na amostra.

Tabela 3.3 Exemplos de pares de palavras para distâncias máximas e mínimas

	X	**Y**
MínD1	profissionalizadas	profissionalizaram
MínD2	procurador	procuradora
MínD3	patrulhada	patrulhadas
MínD4	saturada	saturadas
MínDAbr	profissionalizadas	profissionalizaram

	X	**Y**
MáxD1	norteando	sintetizados
MáxD2	profissionalizaram	pública
MáxD3	pagavam	profissionalizaram
MáxD4	abacaxi	caracteristicamente
MáxDAbr	norteando	simplício

A partir da Tabela 3.3, nota-se que cada par de palavras, cujo resultado da função é mínimo, aponta para o mesmo conceito, estando de acordo com o objetivo da função de distância. Nos casos em que o resultado da função é máximo, os termos representam conceitos totalmente diferentes.

3.2.3 Clusterização de Léxico

A ideia do processo de clusterização é que dados de um mesmo grupo possam apresentar mais características em comum entre si do que com dados de outro grupo. O processo de clusterização de dados é relevante para diversas áreas de pesquisa, tais

como mineração de dados, mineração de textos e reconhecimento de padrões (BISHOP, 2006).

Um processo de clusterização deve levar em conta diversos fatores: a representação dos dados, a similaridade entre os mesmos, como avaliar a qualidade do resultado gerado pelo algoritmo, entre outros. A forma como cada um desses fatores é abordada e os parâmetros a serem ajustados inicialmente representam as principais diferenças entre os algoritmos.

Dentre as soluções clássicas para o processo de clusterização, destacam-se os algoritmos por particionamento e os hierárquicos. No primeiro caso, o conjunto de dados é dividido em um número determinado de *clusters* uma única vez. Entretanto, não se conhece *a priori* o número de *clusters*, tornando essa abordagem impraticável para o problema. No segundo, os dados podem ser divididos gradualmente, obtendo-se diversos *clusters* de acordo com a função de distância adotada.

Uma característica do *clustering* hierárquico é o modo de produção das partições: divisivo ou aglomerativo. No divisivo, a primeira partição é formada por todos os elementos do conjunto de dados, e é divida sucessivamente. No aglomerativo, usado neste trabalho, a primeira partição é formada por diversos *clusters* que são unidos até a formação de uma única partição.

3.3 Adaptação de um *stemmer* ao problema de identificação de domínios

Foi testada a hipótese de adaptação de uma solução de *stemming* para o problema de identificação de domínios em sequências primárias de proteínas. O algoritmo construído, (ALVARES, 2005), apresenta o *stemmer* STEMBR, que essencialmente retira sufixos de palavras da língua portuguesa, e será detalhado a seguir:

Em especial, no STEMBR a retirada de sufixos ocorre com base nas seguintes estratégias:

- Tratamento de sufixos por subconjuntos de palavras que possuem a mesma letra final, e;
- Retirada do maior sufixo possível da palavra.

Na primeira estratégia, o algoritmo identifica a letra final de cada palavra a ser processada, e de acordo com tal grupo realiza o procedimento específico para remoção do sufixo.

Na segunda estratégia, uma lista de sufixos, ordenada de forma decrescente de tamanho, foi criada para cada subconjunto de palavras com mesma letra final. Assim, a remoção do sufixo ocorre percorrendo-se a lista de forma sequencial, verificando se o sufixo é subcadeia final da palavra.

Para esta abordagem ser usada nas sequências de proteínas, surge a necessidade de recuperar os *sufixos* de proteínas cadastradas no banco de dados Pfam. Uma vez que no Pfam é possível identificar o domínio de determinada estrutura primária, considerou-se como *sufixo* a subcadeia à direita do mesmo. Assim, uma lista de diversos *sufixos* de proteínas conhecidas foi extraída, subdividida em vinte grupos de acordo com cada último aminoácido e finalmente ordenada por tamanho, de forma decrescente. Assim, a sequência primária pode ser submetida ao processo, de acordo com os seguintes passos:

i) É identificado o último aminoácido da sequência;

ii) O algoritmo verifica se existe sufixo no grupo de mesmo aminoácido final, começando pelo de maior tamanho, contido no final da sequência de proteínas. Caso verdadeiro, um corte é realizado na sequência primária da proteína, e o domínio candidato corresponde à *string* resultante. Caso contrário o processo é repetido para cada elemento do referido grupo.

Capítulo IV – Estudo de Caso

4.1 Preâmbulo

Este capítulo apresenta dois estudos de caso: o primeiro envolve experimento visando avaliar o modelo de *stemming* (ABR) apresentado no capítulo anterior. O segundo, abrange experimento em sequências primárias de proteínas, com validação estatística da proposta de adaptação de um *stemmer* para estimar domínios de proteínas.

4.2 ABR: Estudo de caso

Aqui, além dos *stemmers* instanciados no processo de clusterização usando as distâncias D_4 e D_5, foram executados os testes nos *stemmers* baseados em regras STEMBR e RSLP, os quais foram desenvolvidos específicamente para a língua portuguesa. Os *stemmers* foram comparados utilizando o Método de Paice.

4.2.1 Amostra utilizada

Foi usada amostra contendo 5000 palavras do idioma português brasileiro, extraída de (ALVARES, GARCIA & FERRAZ 2005).

4.2.2 Análise dos resultados

A Tabela 4.1 a seguir exibe os resultados do Método de Paice aplicado aos *stemmers* baseados em regra (STEMBR e RSLP) e aos dois (D_4 e D_5) resultados oriundos da concepção do modelo ABR. São exibidos os resultados do *Overstemming Index* (OI), *Understemming Index* (UI) e *Stemming Weight* (SW).

Tabela 4.1 *Stemmers* baseados em regras e melhores *clusters* formados

Stemmer	**OI**	**UI**	**SW**
STEMBR	8.41×10^{-5}	0.288	2.90×10^{-4}
RSLP	7.13×10^{-5}	0.295	2.40×10^{-4}
D_4	3.52×10^{-5}	0	-
D_5	3.21×10^{-4}	0	-

Cabe observar que:

- Em se tratando de erros de *overstemming*, o *stemmer* D_4 obteve o menor índice;

- No caso dos erros de *understemming*, os *stemmers* D_4 e D_5 não apresentaram tal tipo de erro. Isso ocorre tendo em vista que é gerado somente um *stem* que representa todas as palavras de cada *cluster*.

4.3 Proteínas

Aqui, são apresentados experimentos realizados com famílias de proteínas. Optou-se pelo uso do banco de dados Pfam o qual provê a possibilidade de obtenção das sequências de proteínas para uso local, dada a necessidade de obtenção dos dados manipulando consultas diretamente nesse banco. Detalhes técnicos a respeito de como obter, instalar e recuperar os dados do banco de dados Pfam são encontrados no capítulo referente aos materiais e métodos.

4.3.1 Amostra utilizada

Uma vez preparado o ambiente que contém as proteínas, foram sorteadas três famílias (PF00004, PF00188 e PF00207), cujas quantidades de sequências estão distribuídas conforme tabela 4.2 a seguir:

Tabela 4.2 Famílias sorteadas e quantidade de sequências

Família	Sequências
PF00004	203
PF00188	100
PF00207	113

4.3.2 Critérios usados para apresentação e análise dos resultados

Uma estatística, a qual chamaremos EST, foi usada como critério para análise dos resultados: corresponde ao tamanho domínio candidato em relação ao tamanho da sequência.

4.3.3 Avaliação

Os resultados do experimento estão expressos na figura 4.1 a seguir, por meio de histogramas:

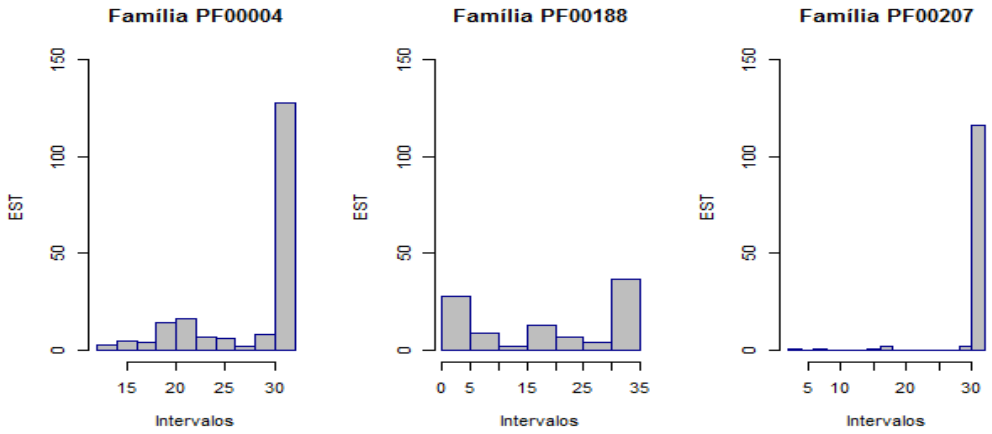

Figura 4.1 Resultado (EST) do experimento por famílias

Na figura 4.2 a seguir é exibido histograma com o resultado da estatística EST aplicada às sequências do experimento. Foram também obtidas a média da estatística μ_{EST}=25,51 e o desvio σ_{EST}=8,37.

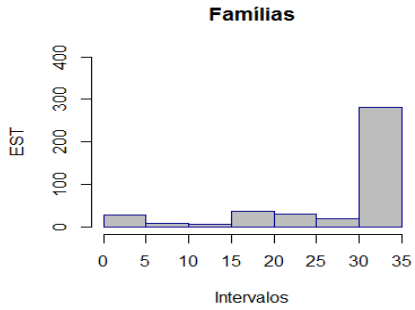

Figura 4.2 Resultado (EST) do experimento

4.3.4 Caracterização da amostra Pfam

Foram sorteadas 40 famílias Pfam (5492 sequências), e em seguida calculada a estatística (ESTPfam), correspondente ao tamanho do domínio Pfam em relação ao tamanho da sequência.

Na figura 4.2 a seguir é exibido histograma com o resultado da estatística ESTPfam aplicada às sequências do experimento. Foram também obtidas a média da estatística $\mu_{ESTPfam}$=52,19 e o desvio $\sigma_{ESTPfam}$=32,93.

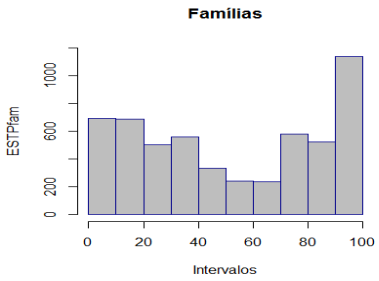

Figura 4.2 Resultado (ESTPfam) em famílias Pfam

A título de ilustração na figura 4.4 a seguir é exibido histograma com o resultado da estatística ESTPfam aplicada a 10 das 40 famílias da amostra.

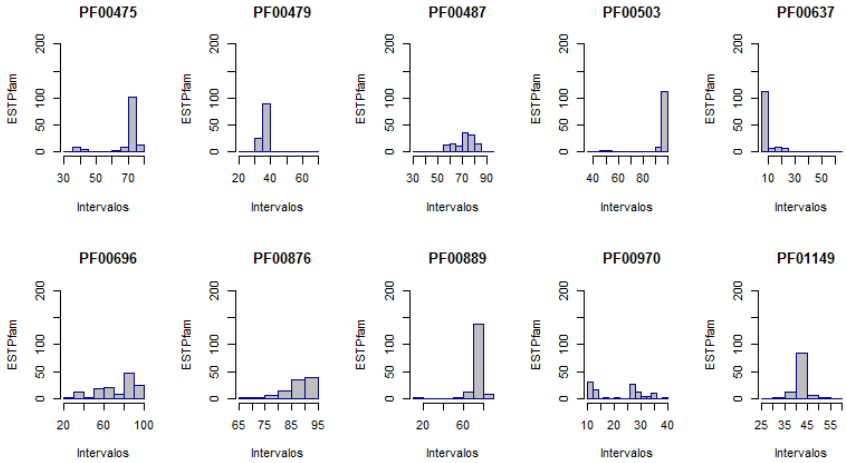

Figura 4.4 Resultado (ESTPfam) em dez famílias Pfam

Na figura 4.5 a seguir é exibido histograma com o resultado da estatística ESTPfam aplicada a outras 10 famílias da amostra.

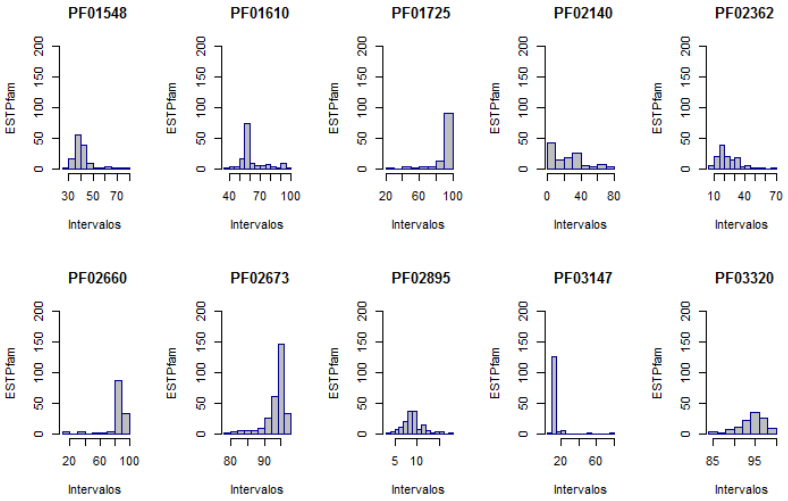

Figura 4.5 Resultado (ESTPfam) em outras dez famílias Pfam

As outras famílias são: PF03457, PF03517, PF03524, PF03595, PF03613, PF03618, PF03880, PF04383, PF04519, PF04760, PF06441, PF07650, PF07908, PF08463, PF09394, PF11740, PF13328, PF13346, PF13419 e PF14363.

4.3.4 Validação Estatística

Nem sempre é possível realizar experimentos científicos em toda a população. No caso das proteínas, até mesmo as que estão catalogadas em bancos de dados representam uma amostra da real população, a qual é difícil de estimar, visto que não se sabe ao certo a quantidade de proteínas existentes (MUSHEGIAN, 2007).

Usualmente, tomam-se amostras aleatórias de uma população para usá-las para obtenção de valores que servirão para estimar e testar hipóteses sobre parâmetros populacionais. Qualquer quantidade obtida de determinada amostra com a finalidade de estimar um parâmetro populacional é chamada estatística amostral.

Considere $x_1,x_2,...,x_n$ variáveis aleatórias independentes e identicamente distribuídas para uma amostra aleatória de tamanho N. A média amostral (ou média da amostra) é definida por:

$$\overline{X} = \frac{x_1 + x_2 + \ldots + x_N}{N} = \frac{\sum x}{N}$$

É natural investigar a distribuição de probabilidade de $\overline{X}$. Estudos de Estatística inferencial demonstram que:

- O valor esperado da média amostral é igual a média populacional;
- Se a população da qual as amostras são estraídas é normalmente distribuída com média μ e variância σ^2 então a média amostral é normalmente distribuída com média μ e variância σ^2/n;
- Suponha que a população da qual as amostras são extraídas tem distribuição de probabildiade com média μ e variância σ^2, mas que não é, necessariamente, a distribuição normal. Então a variável padronizada associada a $\overline{X}$ dada por $Z= \overline{X}-\mu / \sigma/\sqrt{n}$ é assintoticamente normal.

Uma estimativa de algum parâmetro da população dado por um número é denominada estimativa pontual do parâmetro. Já a estimativa dada por dois números entre os quais pode-se considerar que esteja o parâmetro é denominada estimativa por intervalo do parâmetro. Uma afirmação sobre o erro de uma estimativa é comumente referente a sua confiabilidade.

Sejam μ_s e σ_s a média e o desvio-padrão da distribuição amostral de uma estatística S. Caso a distribuição seja aproximadamente normal, podemos esperar que S esteja nos intervalos de $\mu_s - \sigma_s$ a $\mu_s + \sigma_s$, $\mu_s - 2\sigma_s$ a $\mu_s + 2\sigma_s$ ou $\mu_s - 3\sigma_s$ a $\mu_s + 3\sigma_s$ em torno de 68,27%; 95,45% e 99,73% do tempo, respectivamente (SPIEGEL *et al*., 2004). Tais intervalos são denominados intervalos de confiança. Podemos estar confiantes em encontrar μ_s nos intervalos de $S - \sigma_s$ a $S + \sigma_s$, $S - 2\sigma_s$ a $S + 2\sigma_s$ ou $S - 3\sigma_s$ a $S + 3\sigma_s$ em torno de 68,27%; 95,45% e 99,73% do tempo, respectivamente. Estes intervalos são chamados de intervalos de confiança. A porcentagem de confiança é chamada de nível de confiança. Os números 1,96; 2,58 etc. limites de confiança ($S \cong 1{,}96\ \sigma_s$, $S \cong 2{,}58\ \sigma_s$) 95%, 99% para μ_s. $\bar{X} \cong Z_c\ \sigma/\sqrt{n}$ (SPIEGEL *et al*., 2004).

Testes estatísticos são úteis quando há interesse na avaliação de suposições a respeito de um processo. Para isso, há necessidade da formulação de duas proposições mutuamente excludentes. Neste caso, uma proposição afirma uma dada característica do que é estudado; a outra, nega-a.

Por convenção, a característica suposta como verdadeira em relação ao que vai ser avaliado é denominada hipótese nula (indicada por H_0). Já a sua negação, ou seja, proposição tida como verdadeira caso a hipótese nula seja rejeitada, é denominada hipótese alternativa (indicada por H_1).

Cabe observar que, quando realizado um teste de hipóteses, há possibilidade de cometermos erros, comumente denominados erro tipo I e erro tipo II: o primeiro, ocorre quando uma hipótese nula verdadeira é rejeitada; o segundo, quando a hipótese nula é falsa e a mesma não é rejeitada. Erro tipo I somente pode ocorrer quando a hipótese nula é rejeitada. Já o erro tipo II, somente quando a hipótese nula não é rejeitada.

Denomina-se nível de significância à probabilidade máxima com a qual queremos arriscar cometer um erro tipo I. Tal probabilidade em geral é especificada antes de qualquer amostra ser extraída. Em termos práticos, é usual um nível de significância de 0,05 ou 0,01, apesar de outros valores serem usados.

Caso um nível de significância α de 5% seja escolhido, significa que há em torno de cinco chances em cem da hipótese ser rejeitada quando ela deveria ter sido aceita. Sempre que a hipótese nula for verdadeira, significa que estaremos 95% confiantes de que tomamos a decisão correta (podemos estar errados com probabilidade 0,05).

Dada uma estatística S com distribuição normal, com média μ_s e desvio σ_s. Imagine que decidimos rejeitar a hipótese se S for muito pequena ou muito grande. A distribuição da variável padronizada Z é a distribuição normal padrão (média 0,

variância 1). Valores extremos de Z nos levariam à rejeição da hipótese. Vide figura 4.3 a seguir:

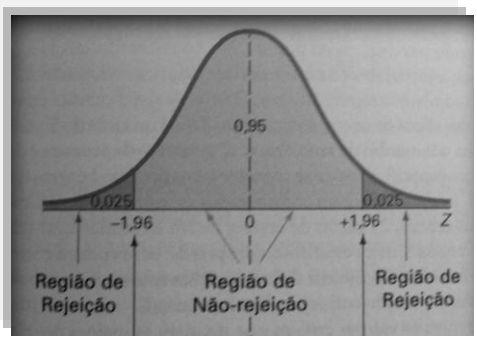

Figura 4.3 Teste Hipótese nível de significância 5% (LEVINE, 2005).

Podemos estar 95% confiantes que, se a hipótese é verdadeira, o escore z de uma estatística amostral verdadeira S estará entre -1,96 e 1,96. Já o conjunto de escores z fora do intervalo de -1,96 e 1,96 constitui o que é denominado por região crítica. O outro conjunto é denominado região de aceitação.

Caso estejamos interessados em valores extremos de um dos lados em uma cauda da distribuição, os testes são denominados unicaudais ou unilaterais. Valores críticos de z para testes unicaudais com nível de significância 0,05 são -1,645 ou 1,645. Valores críticos de z para testes bilaterais com nível de significância 0,05 são -1,96 ou 1,96.

Há três aspectos-chave na metodologia clássica do teste de hipóteses (LEVINE, 2005):

- H_0 sempre se refere a um valor especificado no parâmetro da população (tal como μ) e não a uma estatística da amostra (tal como $\overline{X}$);
- A hipótese nula sempre contém um valor de igualdade com relação ao valor especificado do parâmetro da população;
- A hipótese alternativa nunca contém um sinal de igualdade com relação ao valor especificado do parâmetro da população.

Cabe ressaltar que deixar de rejeitar H_0 não significa provar que ela é verdadeira. é possível concluir apenas que não existem evidências suficientes para garantir sua rejeição. É necessário estudar a distribuição estatística do teste, com base em determinado resultado da amostra uma vez que a distribuição da amostragem para a estatística do teste frequentemente segue uma distribuição padronizada ou a distribuição t. A região de rejeição corresponde aos valores da estatística do teste que são improváveis de ocorrer considerando que a hipótese nula seja verdadeira.

O poder de um teste é uma curva característica de operação. Mostra a probabilidade de erro tipo ii. Fornecem indicações da capacidade dos testes de minimizar erros tipo ii. A probabilidade de cometer um erro tipo ii depende da diferença entre o valor da hipótese e os verdadeiros valores dos parâmetros da população. Deve-se lembrar que um erro tipo ii ocorre caso a hipótese nula não seja rejeitada, quando de fato ela é falsa e deveria ser rejeitada. Caso a diferença seja grande, a probabilidade de ocorrer erro tipo ii (β) será pequena. O poder do teste (1 - β) representa a probabilidade de que seja rejeitada a hipótese nula quando de fato ela for falsa e deva ser rejeitada.

Considere as hipóteses nula e alternativa, conforme a seguir:

H_0 - "A técnica de *stemming* tem acurácia para identificar domínios de proteínas".

H_1 - "A técnica de *stemming* não possui acurácia para identificar domínios de proteínas".

A realização do teste de acordo com a metodologia clássica do teste de hipóteses (LEVINE, 2005), exige os seguintes parâmetros para o cálculo do valor do escore z padronizado:

- Parâmetro populacional (média) μ o qual será calculado e especificado na hipótese nula
- Desvio-padrão populacional σ
- Nivel de significância α
- Parâmetro populacional (média) $\bar{X}$ obtido da amostra
- Número *n* de elementos da amostra

A tabela 4.3 a seguir mostra os valores obtidos para a excução do teste:

Tabela 4.3 Valores usados no teste estatístico

μ	σ	α	$\bar{X}$	***n***
52,19	8,37	0,05	25,51	416

Dados os parâmetros acima, caso ocorra $-1{,}96 \leq z_{calc} \leq 1{,}96$, a hipótese nula será aceita. Caso contrário, a hipótese nula será rejeitada. Uma vez que $z_{calc} = \bar{X}-\mu \ / \ \sigma/\sqrt{n}$ tem-se: $z_{calc} = (25{,}51-52{,}19) / (8{,}37/\sqrt{416}) = -65{,}01$

Dado que $z_{calc} = -65{,}01$, tal valor está localizado na região crítica. A conclusão do teste é que ao nível de significância de 5% a hipótese nula " A técnica de *stemming* tem acurácia para identificar domínios de proteínas " deve ser rejeitada.

O resultado da função poder, dada por 1-β, está expresso na figura 4.4 a seguir:

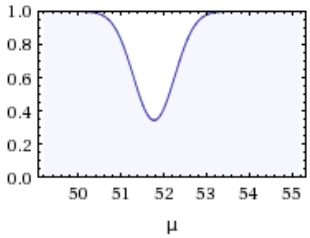

Figura 4.3 Função poder do teste

Capítulo V – Conclusões

A interdisciplinaridade entre Biologia, Computação e Linguagens, aponta para um especial e interessante foco de estudo, uma vez que é interessante a ideia de aplicar soluções de uma área em problemas de outra. No caso das proteínas, foi considerado que a mesma funciona como um idioma, na medida em que subcadeias de proteínas possuem significado biológico relevante.

Vale ressaltar que métodos linguísticos usados em problemas de Bioinformática têm sido aplicados com sucesso há cerca de 20 anos (VERETNIK *et al.*, 2009). Também convém esclarecer que problemas da Biologia, em especial os da Biologia Molecular, são de natureza dinâmica e bastante complexa.

A opção por diminuir a complexidade associada a determinado problema tem sido usada em diversos ramos do conhecimento, dada a esperada facilidade de entendimento do mesmo a partir de representações mais simples do objeto de estudo. Foi comprovado que a técnica de *stemming* não possui acurácia para encontrar domínios de proteínas. Os dados aqui apresentados no teste de hipóteses indicam a falta de acurácia das técnicas de *stemming* para identificar domínios de proteínas, e desaconselhar trabalhos semelhantes.

Referências Bibliográficas

ALBERTS, B., BRAY, D., HOPKIN, K., JOHNSON, A., LEWIS, J., RAFF, M., ROBERTS, KEITH., WALTER, P. Fundamentos da Biologia Celular.Artmed, 2011

ALLAN, J., KUMARAN G. 2003. "Stemming in the language modeling framework". In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in information retrieval*.ACM Press, pp. 455-456.

ALVARES, R. V., 2005. *Investigação do Processo de Stemming na Língua Portuguesa*. Dissertação de Mestrado, Universidade Federal Fluminense, Niterói, RJ, Brasil.

ALVARES, R. V.,GARCIA, A.C.B. "Evaluating stemmers for the Portuguese Language". In *XIII Jornadas Chilenas de Ciencias de la Computación - VI Workshop de Inteligencia Artificial WAI 2005*, Valdivia, Chile. Proceedings of the XIII Jornadas Chilenas de Ciencias de la Computación, v. 1. pp. 1-6.

ALVARES, R. V.,GARCIA, A. C. B., FERRAZ, I. N. 2005. "STEMBR: a Stemming Algorithm for the Brazilian Portuguese Language". In *EPIA 2005 - 12th Portuguese Conference on Artificial Intelligence*, Covilhã, Portugal. v. 3808. pp. 693-701.

ALVARES, R.V., MONDAINI, R.P. 2010. "Um Estudo sobre Funções de Distância entre Palavras na Língua Portuguesa" In *XXXIII Congresso Nacional de Matemática Aplicada e Computacional CNMAC, Modelagem Matemática e Aplicações*. Poster. pp. 1742-1743.

BACCHIN, M., FERRO, N., MELUCCI M. 2002. "University of Padua at CLEF 2002: Experiments to evaluate a statistical stemming algorithm". In *Working Notes for CLEF 2002: Cross-Language Evaluation Forum Workshop*, pp. 161-168.

BISHOP, C. M., 2006, Pattern Recognition and Machine Learning (Information Science and Statistics). Springer-Verlag New York, Inc. Secaucus, NJ, USA.

CHAVES, M.S. 2003. "Um estudo e apreciação sobre algoritmos de stemming para a língua portuguesa". In IX *Jornadas Iberoamericanas de Informática*, Cartagena de Indias, Colômbia.

DE MADARIAGA, R. S., DEL CASTILLO, J. R. F., HILERA, J. R. 2005. "A generalization of the method for evaluation of stemming algorithms based on error counting". In *String processing and information retrieval*. International conference Nº 12, Buenos Aires, Argentine. pp. 228-233.

FRAKES, W.B., FOX C.J. 2003. "Strength and similarity of affix removal stemming algorithms", In *SIGIR Forum*, Vol. 37, pp. 26-30.

FULLER, M., ZOBEL, J., 1998. "Conflation-based Comparison of Stemming Algorithms". In *Proceedings of the Third Australian Document Computing Symposium*. Sydney, Australia. pp. 8-13.

GOLDSMITH J.A., HIGGINS D., SOGLASNOVA S. 2000. "Automatic Language-Specific Stemming in Information Retrieval". In *Workshop of Cross-Language Evaluation Forum on Cross-Language Information Retrieval and Evaluation*. pp.273-284.

HAFER M.A., WEISS S.F., 1974. "Word segmentation by letter successor varieties", *Information Storage and Retrieval*, 10 (11-12), pp. 371-385.

HAMMARSTRON, H. "Poor Man's Stemming: Unsupervised Recognition of Same-Stem Words" In *Information Retrieval Technology, Third Asia Information Retrieval Symposium*. Singapore. pp. 323-337.

HULL, D.A. & GRENFENSTETTE, G., 1996. "A Detailed Analysis of English Stemming Algorithms". Xerox Tech Report.

KANTROWITZ M., MOHIT B., MITTAL V., 2000. "Stemming and its effects on TFIDF ranking" In *Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval*, Athens, Greece. pp. 357-359.

LEVENSTEIN, V.I.: "Binary codes capable of correcting deletions, insertions and reversals". In Commun. 1966. ACM 27,4, pp. 358-368.

LEVINE, D. M., BERENSON, M. L. e STEPHAN, D. Estatística: Teoria e Aplicações usando o Excel. Rio de Janeiro: LTC, 2005.

LOVINS, J., 1968. "Development of a Stemming Algorithm", *Mechanical Translation and Computational Linguistics*, pp. 22-31.

LUCCA, J. L., NUNES, M. G. V. 2002. "Lematização versus Stemming". São Carlos. ICMC-USP. Relatório Técnico.

MAJUMDER, P., MITRA, M., PARUI, S. K., KOLE, G., MITRA, P., DATTA, K. 2007. "YASS: Yet another suffix stripper". In *ACM Transactions on Information Systems(TOIS)*. DOI=10.1145/1281485.1281489

MAYFIELD, J., MCNAMEE, P. 2003. "Single n-gram stemming". In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in information retrieval*. ACM Press, pp. 415-416.

MCNAMEE, P., NICHOLAS, C., MAYFIELD, J. 2008. "Don't have a stemmer?: be un+concern+ed". In *Proceedings of the 31st Annual international ACM SIGIR Conference on Research and Development in information Retrieval*. Singapore. ACM, pp. 813-814.

MELUCCI M., ORIO N. 2003. "A Novel Method for Stemmer Generation Based on Hidden Markov Models". In *Proceedings of Conference on Information and Knowledge Management*. New Orleans, LA, ACM Press. pp. 131-138.

METZLER, D., DUMAIS, S., MEEK, C. 2007. "Similarity Measures for Short Segments of Text". In Advances in Information Retrieval. Springer Berlin, Heidelberg. pp. 16-27.

MOTOMURA, K., NAKAMURA, M., OTAKI, J. M. 2013. *"A frequency-based linguistic approach to protein decoding and design: Simple concepts, diverse applications, and the SCS Package",* In *Computational and Structural Biotechnology Journal.* v. 5: e201302010. doi: http://dx.doi.org/10.5936/csbj.201302010

MUSHEGIAN, A. R., 2007. *Foundations of Comparative Genomics*. Elsevier Academy Press. MA. USA.

NAVATHE, S. B., ELMASRI, R., 2010. *Sistemas de banco de dados*. 6 ed. São Paulo. Pearson Editora.

ORENGO, V. M., HUYCK C. 2001. "*A stemming algorithm for the portuguese language*". e. In Proceedings of Eighth Symposium on String Processing and Information Retrieval (SPIRE 2001), pp. 186-193, Laguna de San Raphael, Chile, November 2001.

ORENGO, V. M., BURIOL, L. S., COELHO, A. R. 2007. "A Study on the Use of Stemming for Monolingual Ad-Hoc Portuguese Information Retrieval". In *Evaluation of Multilingual and Multi-Modal information Retrieval: 7th Workshop of the Cross-Language Evaluation Forum, CLEF 2006, Eds. Lecture Notes In Computer Science*, vol. 4730. Springer-Verlag, Berlin, Heidelberg, pp. 91-98.

PAICE, C. D., 1994. "An evaluation method for stemming algorithms". In *SIGIR*: *Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval*. Springer-Verlag, New York, Inc., pp. 42-50.

PENG, F., AHMED, N., LI, X., LU, Y. 2007. "Context sensitive stemming for web search". In *Proceedings of the 30th Annual international ACM SIGIR Conference*

on Research and Development in information Retrieval. ACM, New York, NY, pp. 639-646.

PORTER, M. F. 1997. "An algorithm for suffix stripping". Morgan Kaufmann Publishers
Inc. San Francisco, CA, pp. 313-316.

POPOV, O., SEGAL, D. M. & TRIFONOV, E. N. Linguistic complexity of protein sequences as compared to texts of human languages. 1996. Biosystems 38, pp. 65–74.

PUNTA M., COGGILL P.C., EBERHARDT R.Y., MISTRY J., TATE J., BOURSNELL C., PANG N., FORSLUND ., CERIC G., CLEMENTS J., HEGER A., HOLM L., SONNHAMMER E.L.L, EDDY S.R., BAEMAN R.D. The Pfam protein families database. 2012. Nucleic Acids Research. Database Issue 40:D290-D301.

RILOFF, E., 1995. “Little words can make a big difference for text classification”. In SIGIR '95: *Proceedings of the 18th annual international ACM SIGIR conference on Research and development in information retrieval*, ACM Press, pp. 130-136.

SEBASTIANI, F. 2002. "Machine learning in automated text categorization". ACM Computing Surveys. pp 1-47.

SETUBAL, J. C. & MEIDANIS, J., 1997, Introduction to Computational Molecular Biology. Boston, PWS Publishing Co.

SHARMA, D. 2012. "Stemming Algorithms: A Comparative Study and their Analysis". International Journal of Applied Information Systems. Published by Foundation of Computer Science, New York, USA. pp 7-12.

SILVA G., OLIVEIRA C. 2003. "The Implementation and Evaluation of a Lexicon-Based Stemmer". In *String Processing and Information Retrieval.* pp. 266-276.

SPIEGEL, M.L., SCHILLER, J. SRINIVASAN, R.A. 2004.Teoria e problemas de probabilidade e estatística - Coleção Schaum. 2 Ed. Bookman. Porto Alegre. RS.

SRINIVASAN, S., THAMBIDURAI, P. 2006. "STANS Algorithm for Root Word Stemming". In *Information Technology Journal*. Volume 5. pp. 685-688.

STEIN, B., POTTHAST, M. 2007. "Putting Successor Variety Stemming to Work". In *Proceedings of the 30th Annual Conference of the Gesellschaft für Klassifikation*. pp. 367-374.

TARS, A., 1976. "Stemming as a system design consideration". In *SIGIR Forum* 11, pp. 9-16.

UYAR, A. 2009. "Google stemming mechanisms". In *Journal of Information Science*. ACM. pp. 499-514.

VERETINIK S., GU J., WODAK S. 2009. Identifying Structural Domains in Proteins. Structural Bioinformatics, Second Edition. John Wiley & Sons, Inc.

WATSON J.D. & CRICK F.H.C. "Molecular Structure of Nucleic Acid: a Structure for Deoxyribose Nucleic Acid." Nature 171, 737, 1953.

WHEELAN, S.J., BAUER, A., BRYANT, S.H. 2000. "Domain size distributions can predict domain boundaries". Bioinformatics. V. 16 pp. 613-618.

WILLIAM, W. C. 2000. "Data integration using similarity joins and a word-based information representation language", *ACM Transactions on Information Systems (TOIS)*, v.18 n.3, pp. 288-321.

WILLIAN B.C., JOHN M.T. 1994. "N-gram-based text categorization", In Proceedings of SDAIR-94, 3rd Annual Symposium on Document Analysis and Information Retrieval. pp. 161-175.

XAVIER, B.M.,SILVA, A.D., GOMES, G.R.R. 2013. Análise comparativa de algoritmos de redução de radicais e sua importância para a mineração de texto. Pesquisa Operacional para o Desenvolvimento, v. 1, pp 84-99.

XU J., CROFT, W.B., 1998. "Corpus-Based Stemming using Co-occurrence of Word Variants". In ACM TOIS, vol. 16, no. 1, pp. 61-81, Computer Science Technical Report TR96-67.

ZIVIANI, N., RIBEIRO-NETO. B. "Text Operations". In: Baeza-Yates, R. and Ribeiro-Neto, B. *Modern Information Retrieval*, Chapter 7, Addison-Wesley, 1999, pp. 163-190.

Materiais e Métodos

Este apêndice aborda sistematicamente procedimentos usados para extrair informações de proteínas do banco de dados Pfam. Para isso, introduz o modelo relacional de banco de dados. Em seguida, aborda uma introdução à linguagem SQL, contemplando exemplos para extração de dados de proteínas.

Modelo relacional e ferramentas.

Banco de dados relacionais armazenam os dados de determinado domínio em estruturas de dados simples, denominadas tabelas. Os primeiros SGBDs surgiram na década de 1970, e desde então sua utilização tem se tornado cada vez mais frequente quando se trata de aplicações que trabalham manipulado grandes volumes de dados. Exemplos de SGBDs utilizados hoje em dia: DB2, Oracle, Postgre SQL, SQL Server, MySQL, entre outros. Este último, é de domínio público, e tem sido utilizado pelo Pfam para gerência e armazenamento dos dados do portal. Por esta razão será detalhado neste trabalho.

O modelo relacional usa o conceito de *relação* matemática, que é semelhante a uma tabela com valores, como seu ponto de partida. Sua base teórica reside em uma teoria de conjunto e lógica de predicado de primeira ordem. Tal modelo será detalhado, após a apresentação do ambiente de experimentos usados na tese, bem como os procedimentos usados para disponibilização dos dados para os experimentos.

Ambiente para os experimentos: MySQL, hardware e Toad for MySQL.

O SGBD MySQL é um *software* livre, e tem sido usado por milhares de aplicações que funcionam na *web*. A versão usanda durante o desenvolvimento do trabalho (5.6), pode ser obtida no endereço http://dev.mysql.com/downloads/mysql/, e está disponível para diversas plataformas de sistemas operacionais, tais como: Linux, Windows, Mac OS X, entre outros.

Os experimentos desta tese foram executados localmente em uma máquina com a seguinte configuração: processador I7, 750 GB de HD e 16GB de memória principal, com o sistema operacional Windows 7. A versão do MySQL utilizada foi a *Community Server* 5.6.14. Após baixar o aplicativo, a instalação padrão segue somente os passos indicados pelo arquivo instalador do programa.

No entanto, uma vez instalado o SGBD, para facilitar o trabalho operacional de estudo e manipulação dos recursos do banco, é conveniente o uso de uma ferramenta

dotada de recursos gráficos para administração do banco e execução das tarefas. Neste caso, foi escolhida a ferramenta livre *Toad for MySQL*, versão 6.7, disponível em http://www.quest.com/toad-for-mysql/. A instalação padrão, a exemplo do MySQL, também segue somente os passos indicados pela ferramenta. A figura 7.1 a seguir é referente à tela do *Toad for MySQL*, já com o banco de dados (PROT) instalado.

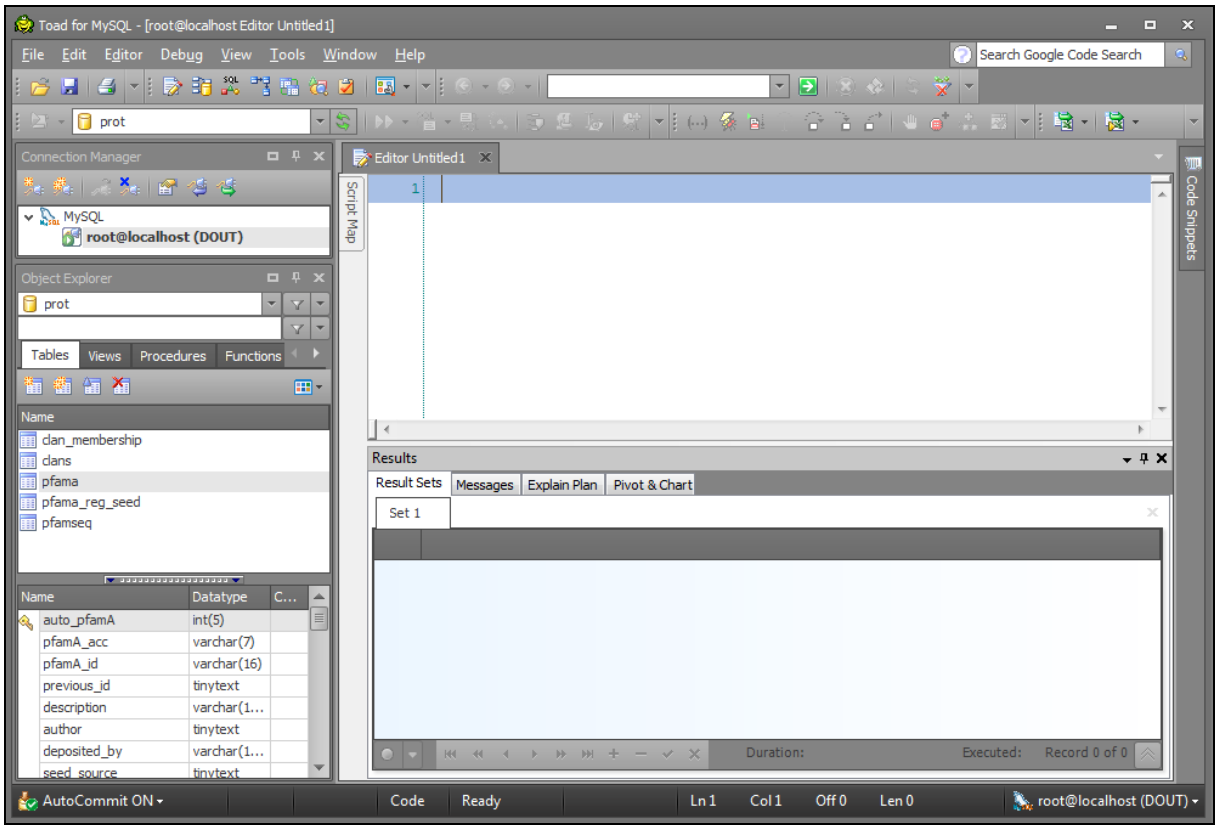

Figura 7.1 Tela da ferramenta Toad for MySQL

Os procedimentos computacionais referentes ao banco de dados são executados no editor de comandos disponível na ferramenta, conforme mostrado na figura 7.1. Os passos para disponibilizar os dados do Pfam na instalação local do MySQL estão detalhados a seguir.

Disponibilizando dados do Pfam no MySQL.

Aqui, são apresentados procedimentos para disponibilizar os dados do Pfam no MySQL para uso local. Tais tarefas correspondem à execução de procedimentos codificados na linguagem SQL (linguagem de consulta estruturada), a qual possui comandos para diversas finalidades. As tarefas de interesse no contexto deste trabalho são:

i) definição da estrutura do banco de dados;

ii) inserção ou carga de dados, e;

iii) recuperação de informações.

A empresa IBM desenvolveu a versão original da linguagem SQL, inicialmente chamada Sequel, no início da década de 70. A linguagem Sequel evoluiu e seu nome foi mudado para SQL (*Structured Query Language*). Atualmente, diversos produtos aceitam a linguagem SQL, a qual tornou-se a linguagem padrão de banco de dados relacional.

Em 1986, o ANSI (*American National Standards Institute*) e a ISO (*International Organization for Standardization*) publicaram um padrão SQL, denominado SQL-86. Em 1989, foi publicado pela ANSI um padrão estendido da linguagem, denominado SQL-89. Em seguida, vieram a SQL-92, SQL-1999 e a SQL-2003. A linguagem possui comandos para diversos propósitos, tais como:

i) Definição de dados (DDL): serve para definição, exclusão e alteração dos componentes do banco, tais como tabelas e relacionamentos;

ii) Manipulação de dados (DML): serve para consulta, inserção, modificação e exclusão de dados;

iii) Integridade: são especificadas restrições as quais os dados armazenados têm que satisfazer. Por exemplo, atualizações que violam as restrições de integridade são proibidas.

Os arquivos de dados do Pfam, necessários ao desenvolvimento do trabalho, estão listados e detalhados a seguir:

-**pfamseq** (5.93 GB compactados): contém as sequências de proteínas. Há informações sobre o identificador da sequência, sua estrutura primária, seu tamanho, entre outras;

-**pfama** (6,49 MB compactados): contém informações sobre as famílias de proteínas. Há informações sobre o identificador da família, autor, anotação da família, entre outras;

-**pfama_reg_seed** (12,11 MB compactados): contém informações sobre os domínios das proteínas. Associa uma sequência de aminoácidos a uma família. Há também os limites (início e fim) do domínio;

-**clans** (59,9 KB compactados): contém informações sobre os clans de famílias;

-**clan_membership** (16,6 KB compactados): contém informações sobre as famílias associadas aos clans.

Uma vez instalados o MySQL e a ferramenta para administração do banco, os seguintes procedimentos são necessários para disponibilizar os dados do Pfam localmente. Leva-se em consideração que os comandos a seguir estão sendo executados na tela de comandos da ferramenta de administração *Toad for MySQL*:

i) Criação do banco de dados: para criar um banco de dados denominado PROT, o seguinte comando deverá ser executado:

```
/*
Cria um banco de dados denominado PROT.
*/
CREATE DATABASE PROT
```

ii) Criação das tabelas: os comandos a seguir servem para criar as tabelas no banco PROT.

```
/*
Torna o banco de dados PROT o banco de trabalho.
 */
USE PROT

/*
Tabela  pfamseq.sql
*/
CREATE TABLE `PFAMA` (
  `AUTO_PFAMA` INT(5) NOT NULL AUTO_INCREMENT,
  `PFAMA_ACC` VARCHAR(7) NOT NULL,
  `PFAMA_ID` VARCHAR(16) NOT NULL,
  `PREVIOUS_ID` TINYTEXT,
  `DESCRIPTION` VARCHAR(100) NOT NULL,
  `AUTHOR` TINYTEXT NOT NULL,
  `DEPOSITED_BY` VARCHAR(100) NOT NULL DEFAULT 'ANON',
  `SEED_SOURCE` TINYTEXT NOT NULL,
  `TYPE` ENUM('FAMILY','DOMAIN','REPEAT','MOTIF') NOT NULL,
  `COMMENT` LONGTEXT,
  `SEQUENCE_GA` DOUBLE(8,2) NOT NULL,
  `DOMAIN_GA` DOUBLE(8,2) NOT NULL,
  `SEQUENCE_TC` DOUBLE(8,2) NOT NULL,
  `DOMAIN_TC` DOUBLE(8,2) NOT NULL,
  `SEQUENCE_NC` DOUBLE(8,2) NOT NULL,
  `DOMAIN_NC` DOUBLE(8,2) NOT NULL,
  `BUILDMETHOD` TINYTEXT NOT NULL,
  `MODEL_LENGTH` MEDIUMINT(8) NOT NULL,
  `SEARCHMETHOD` TINYTEXT NOT NULL,
  `MSV_LAMBDA` DOUBLE(8,2) NOT NULL,
  `MSV_MU` DOUBLE(8,2) NOT NULL,
  `VITERBI_LAMBDA` DOUBLE(8,2) NOT NULL,
  `VITERBI_MU` DOUBLE(8,2) NOT NULL,
  `FORWARD_LAMBDA` DOUBLE(8,2) NOT NULL,
  `FORWARD_TAU` DOUBLE(8,2) NOT NULL,
  `NUM_SEED` INT(10) DEFAULT NULL,
  `NUM_FULL` INT(10) DEFAULT NULL,
  `UPDATED` TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
  `CREATED` DATETIME DEFAULT NULL,
  `VERSION` SMALLINT(5) DEFAULT NULL,
  `NUMBER_ARCHS` INT(8) DEFAULT NULL,
```

```
  `NUMBER_SPECIES` INT(8) DEFAULT NULL,
  `NUMBER_STRUCTURES` INT(8) DEFAULT NULL,
  `NUMBER_NCBI` INT(8) DEFAULT NULL,
  `NUMBER_META` INT(8) DEFAULT NULL,
  `AVERAGE_LENGTH` DOUBLE(6,2) DEFAULT NULL,
  `PERCENTAGE_ID` INT(3) DEFAULT NULL,
  `AVERAGE_COVERAGE` DOUBLE(6,2) DEFAULT NULL,
  `CHANGE_STATUS` TINYTEXT,
  `SEED_CONSENSUS` TEXT,
  `FULL_CONSENSUS` TEXT,
  `NUMBER_SHUFFLED_HITS` INT(10) DEFAULT NULL,
  PRIMARY KEY  (`AUTO_PFAMA`),
  UNIQUE KEY `PFAMA_ACC` (`PFAMA_ACC`),
  UNIQUE KEY `PFAMA_ID` (`PFAMA_ID`)) ;

/*
TABELA  PFAMA.SQL
*/
CREATE TABLE `PFAMSEQ` (
  `AUTO_PFAMSEQ` INT(10) NOT NULL AUTO_INCREMENT,
  `PFAMSEQ_ID` VARCHAR(12) NOT NULL,
  `PFAMSEQ_ACC` VARCHAR(6) NOT NULL,
  `SEQ_VERSION` TINYINT(4) NOT NULL,
  `CRC64` VARCHAR(16) NOT NULL,
  `MD5` VARCHAR(32) NOT NULL,
  `DESCRIPTION` TEXT NOT NULL,
  `EVIDENCE` TINYINT(4) NOT NULL,
  `LENGTH` MEDIUMINT(8) NOT NULL DEFAULT '0',
  `SPECIES` TEXT NOT NULL,
  `TAXONOMY` MEDIUMTEXT,
  `IS_FRAGMENT` TINYINT(1) DEFAULT NULL,
  `SEQUENCE` BLOB NOT NULL,
  `UPDATED` TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
  `CREATED` DATETIME DEFAULT NULL,
  `NCBI_TAXID` INT(10) UNSIGNED DEFAULT '0',
  `GENOME_SEQ` TINYINT(1) DEFAULT '0',
  `AUTO_ARCHITECTURE` INT(10) DEFAULT NULL,
  `TREEFAM_ACC` VARCHAR(8) DEFAULT NULL,
  PRIMARY KEY  (`AUTO_PFAMSEQ`),
  UNIQUE KEY `PFAMSEQ_ACC` (`PFAMSEQ_ACC`),
  KEY `NCBI_TAXID` (`NCBI_TAXID`),
  KEY `CRC64` (`CRC64`),
  KEY `PFAMSEQ_ID` (`PFAMSEQ_ID`),
  KEY `PFAMSEQ_ARCHITECTURE_IDX` (`AUTO_ARCHITECTURE`),
  KEY `PFAMSEQ_NCBI_CODE_IDX` (`NCBI_TAXID`,`GENOME_SEQ`),
  KEY `PFAMSEQ_ACC_VERSION` (`PFAMSEQ_ACC`,`SEQ_VERSION`),
  KEY `MD5` (`MD5`),
  KEY `PFAMSEQ_TAX_IDX` (`TAXONOMY`(350))) ;

/*
```

```
TABELA  PFAMA_REG_SEED.SQL
*/
CREATE TABLE `PFAMA_REG_SEED` (
  `AUTO_PFAMA` INT(5) NOT NULL DEFAULT '0',
  `AUTO_PFAMSEQ` INT(10) NOT NULL DEFAULT '0',
  `SEQ_START` MEDIUMINT(8) NOT NULL DEFAULT '0',
  `SEQ_END` MEDIUMINT(8) NOT NULL,
  `CIGAR` TEXT,
  `TREE_ORDER` MEDIUMINT(9) DEFAULT NULL,
  UNIQUE KEY `PFAMA_REG_SEED_REG_IDX`
(`AUTO_PFAMA`,`AUTO_PFAMSEQ`,`SEQ_START`,`SEQ_END`),
  KEY `AUTO_PFAMA` (`AUTO_PFAMA`),
  KEY `AUTO_PFAMSEQ` (`AUTO_PFAMSEQ`),
  CONSTRAINT `FK_PFAMA_REG_SEED_1` FOREIGN KEY (`AUTO_PFAMA`)
REFERENCES `PFAMA` (`AUTO_PFAMA`) ON DELETE CASCADE ON
UPDATE NO ACTION,
  CONSTRAINT `FK_PFAMA_REG_SEED_2` FOREIGN KEY
(`AUTO_PFAMSEQ`) REFERENCES `PFAMSEQ` (`AUTO_PFAMSEQ`) ON
DELETE CASCADE ON UPDATE NO ACTION);

/*
TABELA  CLANS.SQL
*/
CREATE TABLE `CLANS` (
  `AUTO_CLAN` INT(4) UNSIGNED NOT NULL AUTO_INCREMENT,
  `CLAN_ACC` VARCHAR(6) NOT NULL,
  `CLAN_ID` VARCHAR(40) NOT NULL,
  `PREVIOUS_ID` VARCHAR(75) DEFAULT NULL,
  `CLAN_DESCRIPTION` VARCHAR(100) DEFAULT NULL,
  `CLAN_AUTHOR` TINYTEXT,
  `DEPOSITED_BY` VARCHAR(100) NOT NULL DEFAULT 'ANON',
  `CLAN_COMMENT` LONGTEXT,
  `UPDATED` TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
  `CREATED` DATETIME DEFAULT NULL,
  `VERSION` SMALLINT(5) DEFAULT NULL,
  `NUMBER_STRUCTURES` INT(8) UNSIGNED DEFAULT NULL,
  `NUMBER_ARCHS` INT(8) UNSIGNED DEFAULT NULL,
  `NUMBER_SPECIES` INT(8) UNSIGNED DEFAULT NULL,
  `NUMBER_SEQUENCES` INT(8) UNSIGNED DEFAULT NULL,
  `COMPETED` TINYINT(1) DEFAULT NULL,
  PRIMARY KEY  (`AUTO_CLAN`),
  UNIQUE KEY `CLAN_ACC` (`CLAN_ACC`),
  UNIQUE KEY `CLAN_ID` (`CLAN_ID`));

/*
TABELA  CLAN_MEMBERSHIP.SQL
*/
CREATE TABLE `CLAN_MEMBERSHIP` (
  `AUTO_CLAN` INT(4) UNSIGNED NOT NULL DEFAULT '0',
  `AUTO_PFAMA` INT(10) NOT NULL DEFAULT '0',
```

```
  UNIQUE KEY `CLANMEM` (`AUTO_CLAN`,`AUTO_PFAMA`),
  UNIQUE KEY `FAMILY` (`AUTO_PFAMA`),
  KEY `AUTO_CLAN` (`AUTO_CLAN`),
  KEY `AUTO_PFAMA` (`AUTO_PFAMA`),
  CONSTRAINT `FK_CLAN_MEMBERSHIP_1` FOREIGN KEY (`AUTO_CLAN`)
REFERENCES `CLANS` (`AUTO_CLAN`) ON DELETE CASCADE ON UPDATE
NO ACTION,
  CONSTRAINT `FK_CLAN_MEMBERSHIP_2` FOREIGN KEY
(`AUTO_PFAMA`) REFERENCES `PFAMA` (`AUTO_PFAMA`) ON DELETE
CASCADE ON UPDATE NO ACTION);
```

iii) Carga de dados: Os comandos a seguir são necessários para transferir os dados dos arquivos de texto para as tabelas do banco de dados.

```
/*
CARGA DE DADOS:  LEVA-SE EM CONSIDERAÇÃO QUE OS DADOS ESTÃO
NA PASTA C:/BD
*/
LOAD DATA INFILE "C:/BD/PFAMA.TXT" INTO TABLE PFAMA
LOAD DATA INFILE "C:/BD/PFAMA_REG_SEED.TXT" INTO TABLE
PFAMA_REG_SEED
LOAD DATA INFILE "C:/BD/PFAMSEQ.TXT" INTO TABLE PFAMSEQ
LOAD DATA INFILE "C:/BD/CLANS.TXT" INTO TABLE CLANS
LOAD DATA INFILE "C:/BD/CLAN_MEMBERSHIP.TXT" INTO TABLE
CLAN_MEMBERSHIP
```

Executados os procedimentos anteriores, o banco de dados torna-se pronto para uso nos experimentos. Uma vez que o MySQL representa os dados e seus relacionamentos sob a estrutura de tabelas, serão mostrados a seguir os fundamentos do modelo relacional, que é base para a construção da maioria dos recursos disponíveis nos sistemas gerenciadores de banco de dados relacionais da atualidade.

Modelo relacional.

Aqui, são apresentados conceitos referentes ao modelo relacional de banco de dados. Por razões didáticas, serão explorados exemplos referentes aos dados do Pfam, usados nos experimentos da tese.

Um banco de dados relacional é estruturado por meio de relações ou tabelas. O termo relação foi utilizado na literatura original sobre a abordagem relacional. Já o termo tabela, é uma terminologia comumente encontrada no meio comercial. Uma tabela corresponde a um conjunto não ordenado de tuplas ou linhas, formado por cabeçalhos de atributos ou campos. Para cada atributo, há um conjunto de valores permitidos, que corresponde ao domínio desse atributo. Um exemplo de tabela (tabela

pfamseq) é apresentado na figura 8.2. No exemplo, a tabela armazena dados sobre as sequências de aminoácidos do banco de dados Pfam.

	auto_pfamseq	pfamseq_id	pfamseq_acc	seq_version	crc64	md5	description	evidence	length
1	1	104K_THEPA	P15711	1	289B4B554A61870E	828ea1caefdf0882c2fc7dfe...	104 kDa microneme/rhoptr...	2	924
2	2	108_SOLLC	Q43495	1	CFBAA1231C3A5E92	45c6e4aaae0c760b5133f6b...	Protein 108	2	102
3	3	DEF_VIGUN	P18646	1	6D72D9D238CF7650	119dbffe7be45367410b874...	Defensin-like protein	3	75
4	4	110KD_PLAKN	P13813	1	B0D7CD175C7A3625	5f6bbcf62cb8fa766d6b747b...	110 kDa antigen (Fragment)	2	296
5	5	11S3_HELAN	P19084	1	A007B6F99D189AB5	219a6cb625d16d7f5905c87...	11S globulin seed storage p...	3	493
6	6	11SB_CUCMA	P13744	1	BCD8A83DD1AED93C	e959b769c2d6bab3b627627...	11S globulin subunit beta	1	480
7	8	128UP_DROME	P32234	2	5B38B09D0C0A92F2	8ebf9607cb9a2f0750a1cf16...	GTP-binding protein 128up	1	368
8	9	12AH_CLOS4	P21215	1	A827DB34DB6C8812	d26d8fd55622ce5c5a04bf0...	12-alpha-hydroxysteroid de...	1	29
9	10	12KD_FRAAN	Q05349	1	E44CACBADE6F3C51	ee73cd9bf7cdd70231c1e16...	Auxin-repressed 12.5 kDa p...	2	111
10	11	12KD_MYCSM	P80438	1	0D19F1F488DB3201	404c634d57120b5b7ba9b07...	12 kDa protein (Fragment)	1	24

Figura 7.2 Tabela pfamseq

Cada linha é composta por diversos campos ou colunas. No exemplo, cada linha da tabela corresponde a uma sequência de proteína e cada campo corresponde a uma informação referente a esta sequência (seu identificador, código Uniprot, descrição, ...). Cada campo é identificado por um nome de campo. Na representação gráfica (Figura 7.2) os nomes de campo são representados no cabeçalho da tabela. No exemplo, os nomes de campo são: auto_pfamseq, pfamseq_id, pfamseq_acc, seq_version, crc64, md5, description, evidence e length.

As colunas da tabela **pfamseq** são: ***auto_pfamseq***, *pfamseq_id*, *pfamseq_acc*, *seq_version*, *crc64*, *md5*, *description*, *evidence*, *length*, *species*, *taxonomy*, *is_fragment*, *sequence*, *updated*, *created*, *ncbi_taxid*, *genome_seq*, *auto_architecture* e *treefam_acc*.

Em termos formais, seja D_1 o conjunto de todos os valores de *auto_pfamseq*, D_2 o conjunto de todos valores de *pfamseq_id*, D_3 o conjunto de todos os identificadores *pfamseq_acc* e assim sucessivamente para cada atributo de **pfamseq**. Qualquer linha de **pfamseq** precisa consistir em uma tupla de $n(v_1,v_2,v_3,...,v_n)$, onde v_1 é um *auto_pfamseq* (ou seja, v_1 está no domínio D_1), v_2 é um valor de *pfamseq_id* (ou seja, v_2 está no domínio D_2), v_3 é um valor de *pfamseq_acc* (ou seja, v_3 está no domínio D_3), e assim sucessivamente. Em geral, **pfamseq** conterá um subconjunto do conjunto de todas as linhas possíveis. Portanto, **pfamseq** é um subconjunto de $D_1 \times D_2 \times D_3 \times ... \times D_{19}$. Em geral, uma tabela de n atributos ou colunas precisa ter um subconjunto de $D_1 \times D_2 \times ... \times D_{n-1} \times D_n$ (NAVATHE & ELMASRI, 2010).

Ainda no contexto de relações, é exigido que o domínio de todos os atributos da mesma sejam atômicos. Um domínio de uma relação é atômico se seus elementos são considerados unidades indivisíveis. Por exemplo, o conjunto dos inteiros é um domínio atômico. O aspecto relevante é como usamos elementos de domínio no banco de dados. O domínio da coluna *pfamseq_id* é atômico.

É necessário que o banco de dados relacional possa distinguir linhas em tabelas. No contexto da tabela **pfamseq**, é necessário que o banco diferencie uma sequência das demais. Para isso, é usado o conceito de chave primária. Uma chave primária, corresponde a uma coluna ou combinação de colunas cujos valores servem para distinguir uma linha das demais no contexto de uma tabela. A chave primária representa o principal meio de identificar tuplas em uma relação. Por exemplo, na tabela **Pfamseq**, a chave primária é a coluna ***auto_pfamseq***. Nenhum par de tuplas ou linhas pode ter o mesmo valor de atributos de chave primária ao mesmo tempo. Além disso, é interessante que a escolha de uma chave primária ocorra de modo que seus valores de atributo nunca, ou raramente sejam modificados.

Ainda no contexto da definição de chave primária, é necessário que a mesma seja mínima, ou seja, todas as suas colunas são efetivamente necessárias para garantir o requisito de unicidade de valores da chave. Por exemplo, alguém poderia considerar a combinação de colunas **auto_pfamseq** e **sequence** como sendo uma chave primária. No entanto, se eliminarmos desta combinação a coluna **sequence,** continuamos frente a uma chave primária. Ou seja, a combinação de colunas **auto_pfamseq** e **sequence** não atende ao princípio da minimalidade, portanto não deve ser considerada uma chave primária.

Outro exemplo de tabela (tabela **pfama**) é apresentado na figura 7.3. No exemplo, a tabela armazena dados sobre as famílias de proteínas do banco de dados Pfam.

	auto_pfamA *	pfamA_acc *	pfamA_id *	previous_id	description *	author *	deposited_by *	seed_source *
1	1	PF00389	2-Hacid_dh	2-Hacid_DH;	D-isomer specific 2-hydrox...	Finn RD, Griffiths-Jones SR	anon	Prosite
2	2	PF00198	2-oxoacid_dh		2-oxoacid dehydrogenases ...	Bateman A, Finn RD, Griffit...	anon	Bateman A
3	3	PF04029	2-ph_phosp		2-phosphosulpholactate ph...	Kerrison ND, Finn RD	anon	COG2045
4	4	PF03171	2OG-FeII_Oxy		2OG-Fe(II) oxygenase superf...	Aravind L	anon	Aravind L
5	5	PF01073	3Beta_HSD		3-beta hydroxysteroid dehy...	Finn RD, Bateman A	anon	Pfam-B_504 (release 3.0)
6	6	PF04419	4F5		4F5 protein family	Bateman A, Wood V	anon	Wood V
7	7	PF03061	4HBT		Thioesterase superfamily	Bateman A	anon	Pfam-B_2758 (release 6.4)
8	8	PF02872	5_nucleotid_C	5_nucleotidaseC;	5'-nucleotidase, C-terminal ...	Bateman A, Griffiths-Jones SR	anon	Pfam-B_1318 (release 3.0)
9	9	PF00003	7tm_3		7 transmembrane sweet-tast...	Sonnhammer ELL	anon	Prosite
10	11	PF01661	Macro	DUF27;A1pp;	Macro domain	Bateman A, Mistry J, Wood V	anon	Pfam-B_434 (release 4.1)

Figura 7.3 Tabela pfama

As colunas da tabela **pfama** são: ***auto_pfamA***, *pfamA_acc*, *pfamA_id*, *previous_id*, *description*, *author*, *deposited_by*, *seed_source*, *type*, *comment*, *sequence_GA*, *domain_GA*, *sequence_TC*, *domain_TC*, *sequence_NC*, *domain_NC*, *buildMethod*, *model_length*, *searchMethod*, *msv_lambda*, *msv_mu*, *viterbi_lambda*, *viterbi_mu*, *forward_lambda*, *forward_tau*, *num_seed*, *num_full*, *updated*, *created*, *version*, *number_archs*, *number_species*, *number_structures*, *number_ncbi*, *number_meta*, *average_length*, *percentage_id*, *average_coverage*, *change_status*, *seed_consensus*, *full_consensus* e *number_shuffled_hits*.

Para diferenciar uma família das demais na tabela **pfama**, o banco de dados utiliza a chave primária representada pela coluna **auto_pfama**.

Normalmente, no modelo relacional cada tabela armazena determinado tipo de objeto e suas características. No entanto, tais objetos não são estanque, e em geral possuem relações com outros objetos. Por exemplo, no contexto das proteínas do Pfam, que é um banco de dados de domínios de proteínas, uma sequência pode pertencer a mais de uma família de proteínas. Sabe-se também que uma família possui diversas sequências associadas. Tal informação, ou associação, é representada por meio de um *relacionamento* entre as tabelas **pfamseq** e **pfama**. Em termos mais formais, *uma família possui até **n** sequências e uma sequência possui até **m** famílias.*

Para manter no banco de dados o *relacionamento* ora apresentado, uma vez que em geral temos **n** e **m** maiores que 1 (um), é necessária a criação de uma tabela intermediária (**pfama_reg_seed**), para armazenar cada ocorrência do relacionamento (associação de uma sequência a uma família ou vice-versa). Tal tabela, possui duas colunas (*auto_pfamseq* e *auto_pfamA*), cujos valores apontam necessariamente para os mesmos valores das colunas que são chave primária das tabelas **pfamseq** (*auto_pfamseq*) e **pfama** (*auto_pfamA*), além das colunas que complementam a informação associada ao relacionamento, neste caso *seq_start*, *seq_end*, *cigar* e *tree_order*. As tabelas relacionadas estão exibidas na figura 7.4 a seguir:

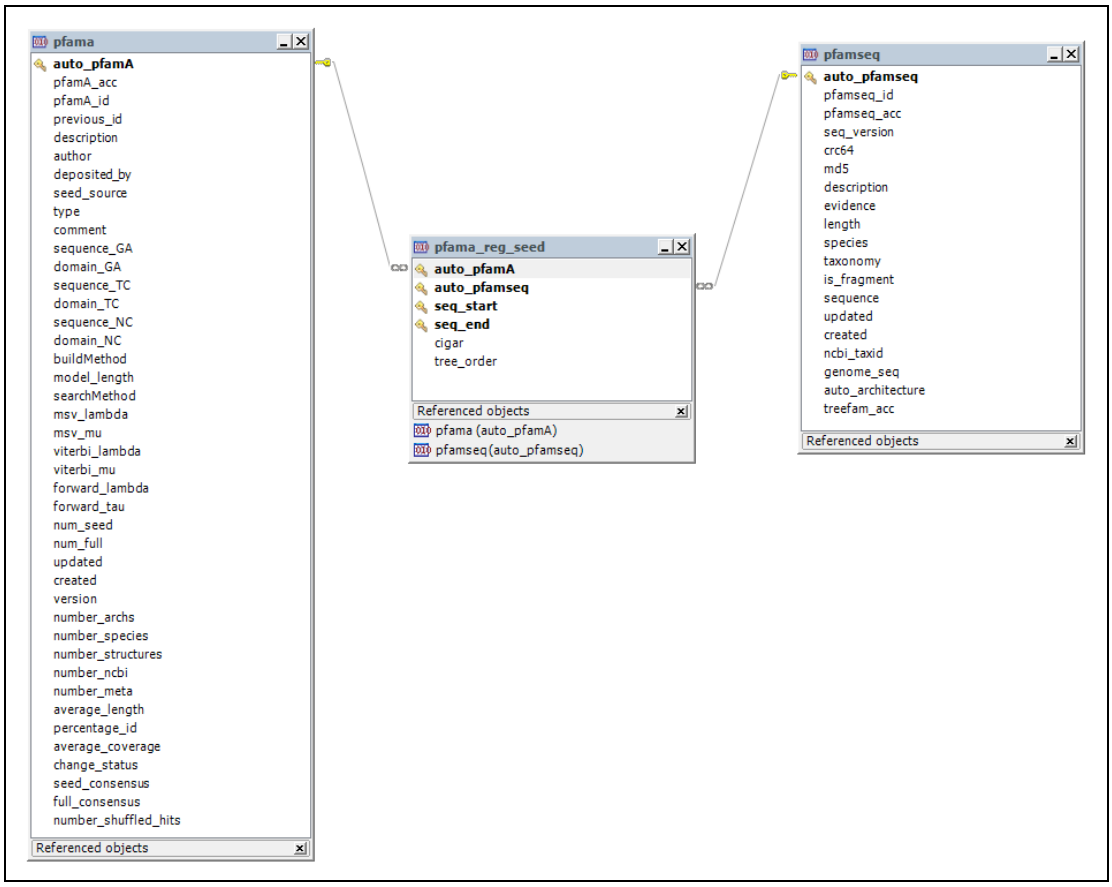

Figura 7.4 Tabela pfama_reg_seed do Pfam e seus relacionamentos

Na tabela **pfama_reg_seed**, além da informação das famílias e sequências (colunas *auto_pfamA* e *auto_pfamseq*, respectivamente), há também a informação sobre o início (*seq_start*) e fim (*seq_end*) do domínio da sequência. Assim, usando as três tabelas apresentadas, é possível recuperar a família, as sequências pertencentes à mesma, bem como a subsequência referente ao domínio associado.

A linha que liga a tabela **pfama** à tabela **pfama_reg_seed** representa o fato de que as mesmas estão relacionadas. O MySQL garante que o valor da coluna *auto_pfamA* da tabela **pfama_reg_seed** seja necessariamente igual a algum valor existente na coluna *auto_pfamA* da tabela **pfama**. Da mesma forma, o MySQL assegura que o valor da coluna *auto_pfamseq* da tabela **pfama_reg_seed** seja necessariamente igual a algum valor existente na coluna *auto_pfamseq* da tabela **pfamseq**. O mecanismo do banco de dados responsável por essa *garantia* é denominado por chave estrangeira.

Outro relacionamento existente no banco de dados Pfam, explorado no contexto deste trabalho, se dá por meio da definição associada a clans de famílias de proteínas. Famílias as quais possuem similaridade são agrupadas em clans.

As colunas da tabela **clans** são: ***auto_clan,*** *clan_acc, clan_id, previous_id, clan_description, clan_author, deposited_by, clan_comment, updated, created, version, number_structures, number_archs, number_species, number_sequences* e *competed.*

A estrutura do Pfam está projetada para armazenar o relacionamento entre famílias e clans, no sentido de que *uma família possui até **n** clans e um clan possui até **m** famílias.* Uma vez que em geral temos **n** e **m** maiores que 1 (um), é necessária a criação de uma tabela intermediária (**clan_membership**), para armazenar cada ocorrência do relacionamento. Tal tabela, possui duas colunas (*auto_clan* e *auto_pfama*), cujos valores apontam necessariamente para as colunas que são chave primária nas tabelas relacionadas **clans** (*auto_clan*) e **pfama** (*auto_pfamA*). As tabelas relacionadas estão exibidas na figura 7.5 a seguir:

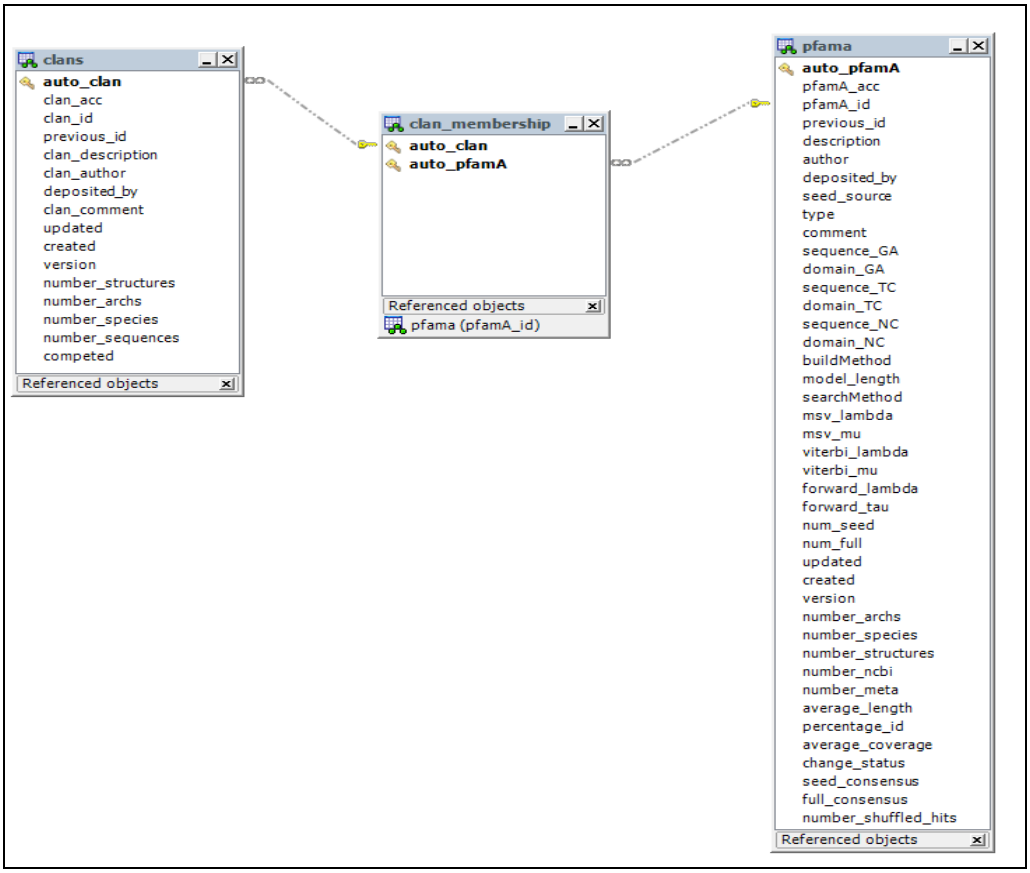

Figura 7.5 Tabela clan_membership do Pfam e seus relacionamentos

Na figura 7.6 a seguir são exibidas as tabelas usadas nos experimentos, bem como as suas relações:

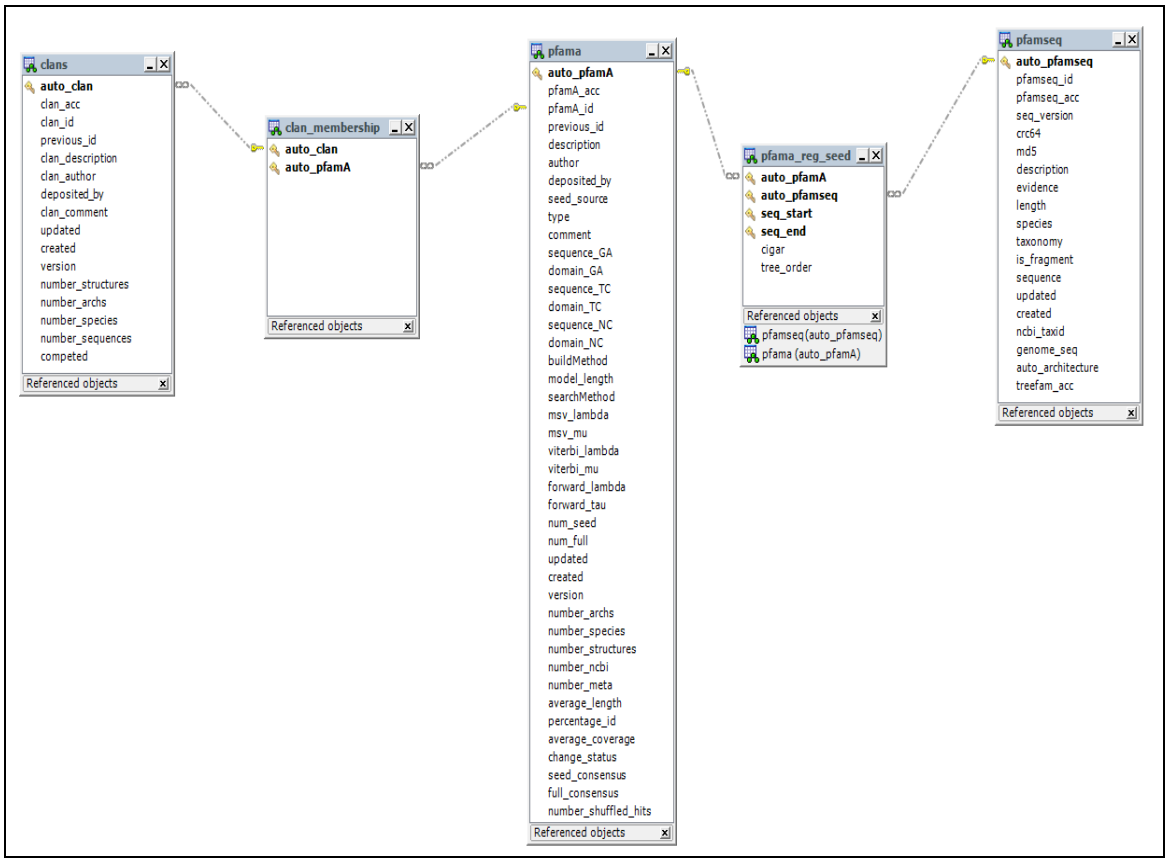

Figura 7.6 Tabelas do Pfam contendo sequências, famílias e clans

As relações, conforme figura 8.6, indicam que um clan (tabela **clans**) pode possuir diversas famílias de proteínas (tabela **pfama**). Uma família pode ter diversas sequências (tabela **pfamseq**). De cada sequência relacionada a uma família é possível recuperar o domínio associado (tabela **pfama_reg_seed**). Serão exibidos procedimentos para recuperação de informações presentes nas tabelas ora apresentadas. Uma vez que tais procedimentos são implementados usando a linguagem SQL, a próxima seção apresenta os comandos mais usuais, necessários para a execução das tarefas.

Sintaxe SQL para execução de consultas.

Em termos de recuperação de informações usando a linguagem SQL, uma sintaxe padrão é apresentada na figura 7.7 a seguir:

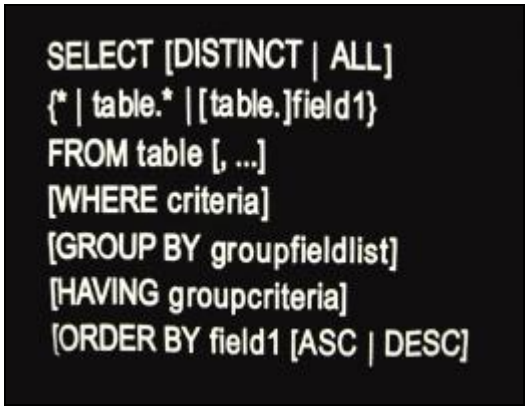

Figura 7.7 Sintaxe SQL para execução de consultas

A figura 7.7 representa a sintaxe padrão SQL utilizada pela maioria dos SGBDs. A seguir, será esclarecido o significado de cada comando:

SELECT [DISTINCT | ALL] {*|table.*|[table].field1}

- Esta linha serve para apresentar ao usuário as colunas (*field*) especificadas;
- As colunas podem existir em alguma tabela, ou podem ser calculadas por meio do uso de alguma função especial (AVG, SUM, COUNT, etc.);
- As linhas resultantes de uma consulta podem apresentar mesmo conteúdo. Utiliza-se DISTINCT para remover linhas repetidas. Para que sejam apresentadas linhas iguais, utiliza-se opção ALL. Na prática, esta opção é utilizada como padrão na maioria dos SGBDs, não sendo necessário explicitá-la;
- table.* indica a recuperação de todas (*) as colunas de determinada tabela da consulta. Para recuperar apenas determinado atributo, substitui-se o * pelo nome da coluna [table].field1.

- FROM table [, ...]

- Nesta linha são declaradas as tabelas envolvidas na consulta;
- Em consultas envolvendo várias tabelas, é necessário indicar os campos que fazem parte do relacionamento para que o SGBD possa recuperar corretamente as informações;

- É interessante notar que o nome de qualquer tabela pode ser virtualmente modificado, principalmente para facilitar a escrita das consultas. Este recurso será utilizado em alguns exemplos mais adiante.

[WHERE criteria]

- Aqui são declarados os mecanismos (conjunto de condições e filtros) necessários à obtenção da informação;
- É possível compor diversos critérios de filtro, usando-se para esse fim uma combinação de operadores lógicos (AND, OR, NOT), subconsultas, operadores de pesquisa em cadeia de caracteres, funções de data, entre outros;
- Em consultas envolvendo várias tabelas, pode-se estabelecer a condição de ligação entre as mesmas.

[GROUP BY groupfieldlist]

- Esta opção é utilizada para agrupar informações em uma consulta;
- Em groupfieldlist declara-se um conjunto de atributos os quais o SGBD considerará como um grupo;
- Quando utilizado em conjunto com funções de agregação (por exemplo, AVG e SUM), os resultados dessas funções são calculados para cada grupo declarado em groupfieldlist.

[HAVING groupcriteria]

- De maneira semelhante à WHERE, esta opção serve para realizar filtros na consulta;
- Os filtros realizados por HAVING são executados após a operação de agrupamento ter sido executada. Neste caso, é possível utilizar funções de agregação como critérios de filtro.

[ORDER BY field1 [ASC | DESC]

- É utilizada para ordenar o resultado da consulta;

- É possível escolher, para cada coluna resultante, o tipo de ordenação (ascendente ou descendente);
- A maioria dos SGBDs utiliza como padrão a ordenação ascendente, neste caso tornando desnecessária a declaração do critério ASC.

Recuperando sequências seeds, famílias e domínios.

Aqui, são detalhados procedimentos usados para recuperação de sequências *seeds* de famílias de proteínas do banco de dados Pfam. Inicialmente serão apresentadas definições usadas no contexto da realização do experimento, finalizando com a apresentação de exemplos.

Recuperando sequências do Pfam.

Foi visto que as sequências encontram-se armazenadas na tabela **pfamseq**. Levando em consideração que o ambiente ora descrito está funcional, é possível, por exemplo, por meio do uso de consultas simples usando a linguagem SQL, recuperar sequências desta tabela. Alguns exemplos:

i) Obter as colunas *auto_pfamseq*, *length*, *pfamseq_acc* e *sequence* das sequências cujo código Uniprot é P53016:

```
SELECT AUTO_PFAMSEQ,PFAMSEQ_ACC,LENGTH, SEQUENCE
FROM PFAMSEQ
WHERE PFAMSEQ_ACC='P53016'
```

Como resultado, tem-se o conteúdo a seguir:

- auto_pfamseq: 1238;

- pfamseq_acc: P53016;

- length: 334;

- sequence:

MEEPMEVDNKRPKVLTWTEKYRPKTLDDIAYQDEVVTMLKGALQGRDLPHLL
FYGPPGTGKTSAALAFCRQLFPKNIFHDRVLDLNASDERGIAVVRQKIQSFSKSS
LGHSHREDVLKLKIIILDEVDAMTREAQAAMRRVIEDFSKTTRFILICNYVSRLIP
PVVSRCAKFRFKSLPAEIQVQRLRTICDAEGTPMSDDELKQVMEYSEGDLRRAV
CTLQSLAPILKSGDDNARNCYLRGSSDSLLISNVCKSILTADVPQIIALTKDITKS
CTGVAFIRRCFQQLMDEDVINDENIGVMGKLVATCEKRILDGCDLENNLLDFLL
TLRETIQ

ii) Recuperar as colunas *auto_pfamseq* e *pfamseq_acc* das sequências que possuem 450 aminoácidos:

```
SELECT AUTO_PFAMSEQ,PFAMSEQ_ACC
FROM PFAMSEQ
WHERE LENGTH=450
```

Como parte dos resultados, são retornados os conteúdos a seguir:

-auto_pfamseq pfamseq_acc

759 Q60474

760 P08913

761 Q01338

762 P18871

763 P22909

iii) Obter a quantidade de sequências que têm 450 aminoácidos.

```
SELECT COUNT(*) AS QUANTIDADE
FROM PFAMSEQ
WHERE LENGTH=450
```

Como resultado, a consulta retorna o valor 18.473. Aqui, foi usada a função *count(*)*, a qual contabiliza o número de linhas resultantes de determinada consulta.

iv) Recuperar as colunas *pfamseq_acc* e *description* das sequências que possuem entre 2000 e 4000 aminoácidos, ordenando o resultado pela coluna *pfamseq_acc*.

```
SELECT PFAMSEQ_ACC, DESCRIPTION
FROM PFAMSEQ
WHERE LENGTH>=2000 AND LENGTH<=4000
ORDER BY PFAMSEQ_ACC
```

Como parte dos resultados, são retornados os conteúdos a seguir:

- *pfamseq_acc description*

A0A062 Polymerase

A0A073 Polyprotein

A0A1F4 Protein eyes shut

A0A379 Protein ycf2

A0A7J4 Putative uncharacterized protein

A0A7N4 Putative uncharacterized protein

A0ABX6 Putative type I polyketide synthase

A0ACH2 Putative polyketide synthase B

Recuperando famílias de proteínas do Pfam.

De forma análoga à obtenção da sequência apresentada na seção anterior, é possível recuperar características das famílias de proteínas. No Pfam, as famílias estão armazenadas na tabela **pfama**.

Alguns exemplos:

i) Obter as colunas *author*, *auto_pfamA*, *description*, *num_full*, *num_seed*, *pfamA_acc* da família PF00004:

```
SELECT  AUTHOR,AUTO_PFAMA,DESCRIPTION,
        NUM_FULL,NUM_SEED,PFAMA_ACC
FROM PFAMA
WHERE PFAMA_ACC='PF00004'
```

Como resultado, tem-se as informações a seguir:

- **author**: Sonnhammer ELL
- **auto_pfamA**: 15
- **description**: ATPase family associated with various cellular activities (AAA)
- **num_full**: 35159
- **num_seed**: 208
- **pfamA_acc**: PF00004

ii) Obter a quantidade de famílias armazenadas na tabela pfamA.

```
SELECT  COUNT(*) AS QUANTIDADE
FROM PFAMA
```

Como resultado, a consulta retorna o valor 13.672.

iii) Obter a quantidade de famílias que têm mais de 100 sequências *seed*.

```
SELECT COUNT(*) AS QUANTIDADE
FROM PFAMA
WHERE NUM_SEED>100
```

Como resultado, a consulta retorna o valor 1.940.

iv) Obter as colunas *author*, *auto_pfamA* e *description* das famílias que têm entre 300 e 500 sequências *seed*.

```
SELECT AUTHOR, AUTO_PFAMA, DESCRIPTION
FROM PFAMA
WHERE NUM_SEED>=300 AND NUM_SEED<=500
```

Como parte dos resultados, são retornados os conteúdos a seguir:

author	auto_pfamA	description
Finn RD	42	AMP-binding enzyme
Sonnhammer ELL	106	C2 domain
Sonnhammer ELL, Finn RD	145	Cyclic nucleotide-binding domain
Dlakic M	346	Endonuclease/Exonuclease/phosphatase family
Bateman A	353	F-box domain
Bateman A	368	FG-GAP repeat

Recuperando domínios seed de uma família.

Recuperar uma família, suas sequências *seeds* e seus domínios associados requer a construção de uma consulta que envolve as três tabelas (**pfamseq**, **pfama_reg_seed** e **pfama**) apresentadas na figura 4.6. Ao construir consultas envolvendo informações presentes em mais de uma tabela, torna-se necessário utilizar um comando de junção (*join*), entre as tabelas envolvidas, além de explicitar a condição da junção, que na maioria das vezes é representada por uma igualdade entre as colunas relacionadas das tabelas envolvidas.

Em consultas envolvendo o comando de junção (*join*), para que as informações sejam recuperadas do banco de dados, SGBD forma internamente uma tabela intermediária cujo número de linhas é correspondente ao resultado do produto cartesiano entre as linhas das tabelas envolvidas. Além disso, tal estrutura contém todas as colunas dessas tabelas. A partir daí, é necessário estabelecer uma condição de filtro (condição da junção) para que somente as linhas de interesse sejam retornadas. Neste caso o interesse é nas linhas cujos valores dos campos correspondentes às chaves primária e estrangeira sejam os mesmos. O comando de junção (*join*) é declarado na cláusula *from*.

Alguns exemplos:

i) Recuperar, para a família PF00004, seu identificador, o código UNIPROT, as sequências de aminoácidos e os domínios de suas sequências *seed*.

```
SELECT PA.PFAMA_ACC, P.PFAMSEQ_ACC, P.SEQUENCE,
SUBSTRING(P.SEQUENCE,PM.SEQ_START,PM.SEQ_END-
PM.SEQ_START+1) AS DOMINIO
FROM     PFAMSEQ P JOIN PFAMA_REG_SEED PM ON
(P.AUTO_PFAMSEQ = PM.AUTO_PFAMSEQ)
   JOIN PFAMA PA ON (PM.AUTO_PFAMA = PA.AUTO_PFAMA)
WHERE    PA.PFAMA_ACC IN('PF00004')
```

Neste caso, a informação sobre o domínio da sequência está contida na tabela **pfama_reg_seed**, nos valores das colunas *seq_start* e *seq_end*. Tais valores representam as posições inicial e final do domínio da sequência. Para que o domínio possa ser recuperado, foi utilizada a função *substring*, a qual recupera uma subcadeia de caracteres a partir de uma coluna. A função requer, nessa ordem, os seguintes parâmetros: coluna, posição inicial e tamanho da subsequência. Para saber o tamanho do domínio, é subtraída a posição final da posição inicial do mesmo.

Por razões de espaço, serão mostrados apenas os três primeiros resultados da consulta.

- pfamA_acc: PF00004

- pfamseq_acc: P53016

- sequence:

MEEPMEVDNKRPKVLTWTEKYRPKTLDDIAYQDEVVTMLKGALQGRDLPHLL
FYGPPGTGKTSAALAFCRQLFPKNIFHDRVLDLNASDERGIAVVRQKIQSFSKSS
LGHSHREDVLKLKIIILDEVDAMTREAQAAMRRVIEDFSKTTRFILICNYVSRLIP
PVVSRCAKFRFKSLPAEIQVQRLRTICDAEGTPMSDDELKQVMEYSEGDLRRAV
CTLQSLAPILKSGDDNARNCYLRGSSDSLLISNVCKSILTADVPQIIALTKDITKS
CTGVAFIRRCFQQLMDEDVINDENIGVMGKLVATCEKRILDGCDLENNLLDFLL
TLRETIQ
-dominio:
LLFYGPPGTGKTSAALAFCRQLFPKNIFHDRVLDLNASDERGIAVVRQKIQSFSK
SSLGHSHREDVLKLKIIILDEVDAMTREAQAAMRRVIEDFSKTTRFILICNYVSR
LIPPVVSRCAKFRFKS

- pfamA_acc: PF00004

- pfamseq_acc: P35249

- sequence:

MQAFLKGTSISTKPPLTKDRGVAASAGSSGENKKAKPVPWVEKYRPKCVDEVA
FQEEVVAVLKKSLEGADLPNLLFYGPPGTGKTSTILAAARELFGPELFRLRVLEL
NASDERGIQVVREKVKNFAQLTVSGSRSDGKPCPPFKIVILDEADSMTSAAQAA
LRRTMEKESKTTRFCLICNYVSRIIEPLTSRCSKFRFKPLSDKIQQQRLLDIAKKE
NVKISDEGIAYLVKVSEGDLRKAITFLQSATRLTGGKEITEKVITDIAGVIPAEKI
DGVFAACQSGSFDKLEAVVKDLIDEGHAATQLVNQLHDVVVENNLSDKQKSII
TEKLAEVDKCLADGADEHLQLISLCATVMQQLSQNC
-dominio:
LLFYGPPGTGKTSTILAAARELFGPELFRLRVLELNASDERGIQVVREKVKNFAQ
LTVSGSRSDGKPCPPFKIVILDEADSMTSAAQAALRRTMEKESKTTRFCLICNYV
SRIIEPLTSRCSKFRFKP

- pfamA_acc: PF00004

- pfamseq_acc: P35600

- sequence:

MQRGIDSFFKRLPAKAKSAEAENGETPSKAPKRRKAVIISSDEDEVVSPPETKKR
KASKTASSEDDVVAATPEPIAKKARNGQKPALSKLKRHVDPTELFGGETKRVIV
PKPKTKAVLEFENEDIDRSLMEVDLDESIKEAAPEKKVHSITRSSPSPKRAKNSS
PEPPKPKSTKSKATTPRVKKEKPAADLESSVLTDEERHERKRASAVLYQKYKN
RSSCLNPGSKEIPKGSPDCLSGLTFVVTGVLESMEREEAESVIKEYGGKVMTVV
GKKLKYLVVGEEAGPKKLAVAEELNIPILSEDGLFDLIREKSGIAKQVKEEKKSP
KKEHSSEEKGKKEVKTSRRSSDKKEKEATKLKYGEKHDIAKHKVKEEHTSPKE
TKDKLNDVPAVTLKVKKEPSSQKEHPPSPRTADLKTLDVVGMAWVDKHKPTSI
KEIVGQAGAASNVTKLMNWLSKWYVNHDGNKKPQRPNPWAKNDDGSFYKA

ALLSGPPGIGKTTTATLVVKELGFDAVEFNASDTRSKRLLKDEVSTLLSNKSLSG
YFTGQGQAVSRKHVLIMDEVDGMAGNEDRGGMQELIALIKDSSIPIICMCNDR
NHPKIRSLVNYCYDLRFQRPRLEQIKGKIMSICFKEKVKISPAKVEEIIAATNNDI
RQSINHIALLSAKEDASQKSGQQVATKDLKLGPWEVVRKVFTADEHKHMSFA
DKSDLFFHDYSLAPLFVQQNYLQVLPQGNKKDVLAKVAATADALSLGDLVEK
RIRANSAWSLLPTQAFFSSVLPGEHMCGHFTGQINFPGWLGKNSKSGKRARLA
QELHDHTRVCTSGSRLSVRLDYAPFLLDNIVRPLAKDGQEGVPAALDVMKDYH
LLREDLDSLVELTSWPGKKSPLDAVDGRVKAALTRSYNKEVMAYSYSAQAGI
KKKKSEAAGADDDYLDEGPGEEDGAGGHLSSEEDEDKDNLELDSLIKAKKRTT
TSKASGGSKKATSSTASKSKAKAKK

- dominio:

ALLSGPPGIGKTTTATLVVKELGFDAVEFNASDTRSKRLLKDEVSTLLSNKSLSG
YFTGQGQAVSRKHVLIMDEVDGMAGNEDRGGMQELIALIKDSSIPIICMCNDR
NHPKIRSLVNYCYDLRFQRP

ii) Obter a quantidade de sequências *seed* pertencentes à família PF00004.

```
SELECT  COUNT(*) AS QUANTIDADE
FROM     PFAMSEQ P JOIN PFAMA_REG_SEED PM ON
(P.AUTO_PFAMSEQ = PM.AUTO_PFAMSEQ)
        JOIN PFAMA PA ON (PM.AUTO_PFAMA =
PA.AUTO_PFAMA)
WHERE   PA.PFAMA_ACC IN('PF00004')
```

Como resultado, a consulta retorna o valor 208.

Recuperando clans e famílias.

Foi visto que um clan corresponde a um conjunto de famílias que possuem similaridade associada. As consultas a seguir detalham procedimentos de recuperação de informações a respeito dos clans de famílias e suas características.

Alguns exemplos:

i) Obter a quantidade de clans armazenadas na tabela clans.

```
SELECT  COUNT(*) AS QUANTIDADE
FROM CLANS
```

Como resultado, a consulta retorna o valor 515.

ii) Obter a quantidade de famílias por clan.

```
SELECT AUTO_CLAN, COUNT(*) AS QUANTIDADE
FROM CLAN_MEMBERSHIP
GROUP BY AUTO_CLAN
ORDER BY COUNT(*)
```

A função *count(*)* retorna a quantidade de linhas da consulta. Uma vez que a consulta do exemplo em questão ocorre diretamente na tabela **clan_membership**, o resultado da função corresponde ao número de famílias por clan. Cabe ressaltar que coluna auto_clan está sendo usada no comando group by, implicando que o cálculo de *count(*)* ocorra para cada instância dessa coluna, ou seja, é calculado o número de linhas em que cada identificador de família aparece na tabela.

Por razões de espaço, serão mostrados apenas os cinco primeiros resultados da consulta.

auto_clan quantidade

195 1
345 2
410 2
447 2
483 2
521 2

iii) Obter a média de famílias por clan.

```
SELECT AVG(QUANTIDADE) AS MEDIA
FROM
(SELECT AUTO_CLAN, COUNT(*) AS QUANTIDADE
FROM CLAN_MEMBERSHIP
GROUP BY AUTO_CLAN)AS TESTE
```

Resultado: 8

iv) Obter os clans cujas famílias têm mais de 30 sequências

```
SELECT DISTINCT CLAN
FROM  ( SELECT CM.AUTO_CLAN AS CLAN,
       P.PFAMA_ACC AS FAMILIA, PM.AUTO_PFAMA,
       COUNT(*) AS QTDE
    FROM   PFAMA P JOIN PFAMA_REG_SEED PM ON
(P.AUTO_PFAMA=PM.AUTO_PFAMA) JOIN CLAN_MEMBERSHIP
CM ON (CM.AUTO_PFAMA=PM.AUTO_PFAMA)
    GROUP BY PM.AUTO_PFAMA
    ORDER BY CM.AUTO_CLAN,COUNT(*)) AS TESTE
WHERE CLAN NOT IN
( SELECT DISTINCT CLAN
  FROM
     (SELECT CM.AUTO_CLAN AS CLAN,
       P.PFAMA_ACC AS FAMILIA,
       PM.AUTO_PFAMA,
       COUNT(*) AS QTDE
    FROM   PFAMA P JOIN PFAMA_REG_SEED PM ON
(P.AUTO_PFAMA=PM.AUTO_PFAMA)
       JOIN CLAN_MEMBERSHIP CM ON
(CM.AUTO_PFAMA=PM.AUTO_PFAMA)
    GROUP BY PM.AUTO_PFAMA
    HAVING COUNT(*)<30
    ORDER BY CM.AUTO_CLAN,COUNT(*))AS TESTE1)
ORDER BY CLAN
```

Como parte dos resultados, tem-se os seguintes clans: 5, 7, 22, 25, 41, 71, 77, 87, 100, 139.

v) Obter o código Uniprot das famílias pertencentes ao clan CL0328.

```
SELECT P.PFAMA_ACC
FROM  CLANS C JOIN CLAN_MEMBERSHIP CM
      ON (C.AUTO_CLAN = CM.AUTO_CLAN) JOIN
      PFAMA P ON (CM.AUTO_PFAMA = P.AUTO_PFAMA)
WHERE CLAN_ACC='CL0328'
```

Como resultado, tem-se as informações a seguir:

PF02628 PF03462 PF01578 PF03188 PF00033

PF01292 PF10348 PF13631 PF14358 PF01794

vi) Obter a quantidade de sequências das famílias que compõem o clan CL0328.

```
SELECT COUNT(*) AS QUANTIDADE
FROM  CLANS C JOIN CLAN_MEMBERSHIP CM
      ON (C.AUTO_CLAN = CM.AUTO_CLAN) JOIN
      PFAMA P ON (CM.AUTO_PFAMA = P.AUTO_PFAMA)
JOIN PFAMA_REG_SEED PR ON (PR.AUTO_PFAMA =
P.AUTO_PFAMA)
WHERE CLAN_ACC='CL0328'
```

Resultado: 948

vii) Obter o código Uniprot das famílias pertencentes ao clan CL0021.

```
SELECT P.PFAMA_ACC
FROM  CLANS C JOIN CLAN_MEMBERSHIP CM
      ON (C.AUTO_CLAN = CM.AUTO_CLAN) JOIN
      PFAMA P ON (CM.AUTO_PFAMA = P.AUTO_PFAMA)
WHERE CLAN_ACC='CL0021'
```

Como resultado, tem-se as informações a seguir (total de 44 famílias):

PF00313 PF00436 PF03459 PF01938 PF01336 PF01588 PF03120 PF02721

PF01796 PF04076 PF01132 PF01176 PF01287 PF03319 PF03919 PF02303

PF04057 PF00181 PF00164 PF00366 PF03870 PF01330 PF00575 PF02765

PF07404 PF07497 PF07717 PF08206 PF08292 PF08402 PF08661 PF09883

PF10447 PF10451 PF11325 PF11967 PF12658 PF12857 PF12869 PF13114

PF13376 PF13509 PF13742 PF14444

viii) Obter a quantidade de sequências das famílias que compõem o clan CL0021.

```
SELECT COUNT(*) AS QUANTIDADE
FROM  CLANS C JOIN CLAN_MEMBERSHIP CM
      ON (C.AUTO_CLAN = CM.AUTO_CLAN) JOIN
      PFAMA P ON (CM.AUTO_PFAMA = P.AUTO_PFAMA)
JOIN PFAMA_REG_SEED PR ON (PR.AUTO_PFAMA =
P.AUTO_PFAMA)
WHERE CLAN_ACC='CL0021'
```

Resultado: 2987

ix) Recuperar os clans cujas famílias têm mais de 30 sequencias, mostrando os clans com maior quantidade de famílias primeiro

```
SELECT CLAN, COUNT(*) AS QTDEFAM
FROM
       (SELECT CLAN,FAMILIA,AUTO_PFAMA,QTDE
       FROM
       ( SELECT CM.AUTO_CLAN AS CLAN,
        P.PFAMA_ACC AS FAMILIA,
        PM.AUTO_PFAMA,
        COUNT(*) AS QTDE
    FROM   PFAMA P JOIN PFAMA_REG_SEED PM ON
(P.AUTO_PFAMA=PM.AUTO_PFAMA)
       JOIN CLAN_MEMBERSHIP CM ON
(CM.AUTO_PFAMA=PM.AUTO_PFAMA)
    GROUP BY PM.AUTO_PFAMA
    ORDER BY CM.AUTO_CLAN,COUNT(*)) AS TESTE
    WHERE CLAN NOT IN
(SELECT DISTINCT CLAN
  FROM
     (SELECT CM.AUTO_CLAN AS CLAN,
        P.PFAMA_ACC AS FAMILIA,
        PM.AUTO_PFAMA,
        COUNT(*) AS QTDE
    FROM   PFAMA P JOIN PFAMA_REG_SEED PM ON
(P.AUTO_PFAMA=PM.AUTO_PFAMA)
       JOIN CLAN_MEMBERSHIP CM ON
(CM.AUTO_PFAMA=PM.AUTO_PFAMA)
    GROUP BY PM.AUTO_PFAMA
    HAVING COUNT(*)<30
    ORDER BY CM.AUTO_CLAN,COUNT(*))AS TESTE1)
ORDER BY CLAN   ) AS FINAL
GROUP BY CLAN
ORDER BY COUNT(*) DESC
```

Como parte dos resultados, tem-se:

clan	qtdefam
328	10
22	10
25	8