Laboratório Biodados
Departamento de Bioquímica e Imunologia
Universidade Federal de Minas Gerais
Av. Antônio Carlos 6627, 31270-901 Belo Horizonte, MG Brasil

Desenvolvimento de um serviço de análise de sequências utilizando um modelo baseado em atributos de resultados de PSI-BLAST

Henrique de Assis Lopes Ribeiro

Tese submetida à banca examinadora designada pelo Colegiado do Programa de Pós-Graduação em Bioinformática da Universidade Federal de Minas Gerais, como parte dos requisitos necessários à obtenção do grau de Doutor em Bioinformática.

Orientador: Prof. José Miguel Ortega, PhD

Belo Horizonte, dezembro de 2013

Agradecimentos

Agradeço em primeiro lugar a meus pais Ana Maria de Assis Lopes Ribeiro e Alfredo Lopes Ribeiro pelo suporte e pelo amor.

Agradeço também a todos os colegas do laboratório biodados, ao Miguel pelo otimismo e auto astral, a Katia, Raquel, Tetsu, Lucas, Ricardo e Marcele pelo apoio e amizade.

Agradeço a todos os meus amigos do CEMFS.

E termino agradecendo a Deus

Sumário

Resumo

O PSI-BLAST é uma das principais ferramentas de busca de homólogos distantes. Este tipo de busca é importante para modelagem molecular, predição de estrutura secundária e anotação funcional de proteínas hipotéticas. No entanto a busca PSI-BLAST é reportada na literatura como tendo uma alta taxa de falsos positivos. Isso ocorre devido ao que é conhecido na literatura com corrupção da PSSM. Esta corrupção se deve em grade parte ao fato de que a PSSM é calculada de maneira não supervisionada.

Neste trabalho nós combinamos PSI-BLAST com técnicas de aprendizado supervisionado que foram capazes de calcular a probabilidade de um resultado estar correto. Para isso nos utilizamos 1200 proteínas de PANTHER para treinar e testar os modelos. Nós selecionamos 800 destas proteínas para treino e 400 para teste. Para isso disparamos PSI-BLAST contra um banco de dados multi-fasta PANTHER-UniProt e monitoramos quais proteínas recuperadas pelo PSI-BLAST eram do mesmo *cluster* PANTHER da *query*, quais eram de *cluster* PANTHER diferente e desconsideramos aquelas que não pertenciam a nenhum *cluster*. Foram criadas neste trabalho diversos preditores (*features*) baseados nas pontuações dos *subjects* (query inclusa pois aparece nos resultados) em cada iteração e com isso foi treinado um ensemble de redes neurais e *random forest* que atingiu 0.94 de AUC nos 400 PSI-BLASTs de teste.

Este modelo foi aplicado a 900 PSI-BLASTs com proteínas de anotação recente onde foram monitoradas a similaridade entre as anotações dos *subjects* e a anotação da *query*. Estes testes acabaram por gerar um modelo de ponderação de importância das anotações e sugestão de anotação baseado em consenso de anotações ao invés de anotação baseado em *best-hit*.

O modelo de PSI-BLAST-ML (PSI-BLAST-Machine Learning) que criamos foi aplicado a proteínas de função desconhecida de quatro micro-organismos e cerca de metade destas proteínas puderam receber alguma sugestão de anotação.

Por fim estes modelos juntamente cum varias outras métricas foram integrados em uma ferramenta web chamada Annothetic (Annotate hypothetical). Esta ferramenta pode ser usada para sugerir função para proteínas hipotéticas de difícil anotação bem como outras aplicações que requeiram a busca de homólogos distantes.

Abstract

PSI-BLAST is one of the main tools for remote homology search. This kind of task is essential for molecular modeling, secondary structure prediction and hypothetical proteins functional annotation. Nevertheless literature reports high rates of false positives in PSI-BLAST search. That is mostly due to the unsupervised way PSI-BLAST calculates the PSSM weights.

In this work we combine PSI-BLAST with supervised machine learn techniques that were able to predict probability of a result being correct. In order to do that 1200 PANTHER's queries were selected and split in two groups: one with 800 were used as training and another of 400 were tests. These queries were submitted to PSI-BLAST against a PANTHER-UniProt multi-fasta database. Then each subject found was evaluated as being from the same cluster as the query, from a different cluster, or as not having a cluster, in which case the subject were discarded. Also 17 features were created based on the subject scores found in each iteration and query size. With these features an ensemble of neural networks and random forest were trained and achieved 0.94 AUC in test.

The 1200 queries were also submitted to BLASTp and a neuronal network model was trained and achieved 0.78 AUC in test. This model only takes 3 features and was proposed as a heuristic for the main model based on PSI-BLAST.

These ML-BLAST (Machine Learn- BLAST) models were applied to 900 recent annotated proteins and subject and query's annotations similarity were compared. These tests happened to generate a model of weighted annotation relevance. And annotation suggestion based on annotation consensus.

ML-BLAST models were also applied to four microorganism's hypothetical proteins and ware able to suggest annotation for about half of them.

These models jointly with a set of other metrics were integrated in a new tool called Annothetic (Annotate Hypothetical). Despite of the name, this tool can be applied not only for proteins annotation but also for any task that require remote similarity search.

Lista de Figuras

Lista de Tabelas

Lista de Abreviações

IA	Inteligência Artificial
ATP	Adenosine TriphosPhate
AUC	Area Under the Curve
BLAST	Basic Local Alignment Search Tool
BLOSUM	BLOcks of Amino Acid SUbstitution Matrix
DNA	DeoxyriboNucleic Acid
DSSP	Define Secondary Structure of Proteins
EBI	European Bioinformatics Institute
EC	Enzyme Comission
GI	General Identifier
GO	Gene Ontology
HMM	Hidden Markov Model
KEGG	Kyoto Encyclopedia of Genes and Genomes
kNN	k-Nearest Neighbors
KO	KEGG Orthology
LCA	Lowest Common Ancestor
MATLAB	MATrix LABoratory
ML	Machine Learn
MLP	MultiLayer Perceptron
MUSCLE	Multiple Sequence Comparison by Log-Expectation
NCBI	National Center for Biotechnology Information
NNF	Non-Negative Factorization
PAM	Point Accepted Mutation
PANTHER	Protein ANalysis THrough Evolutionary Relationships
PDB	Protein Data Bank
Pfam	Protein family
PHP	Personal Hypertext Preprocessor
PIR	Protein Information Resource
PSI-BLAST	Position Specific Iterated BLAST
PSSM	Position Specific Score Matrix
PWM	Position Weight Matrix
RBF	Radial Basis Function
RF	Random Forest
RNA (aprendizado de máquina)	Redes Neurais Artificiais
RNA (biologia)	RiboNucleic Acid
SOV	Segment OVerlap
SVD	Singular Value Decomposition
SVM	Suport Vector Machine

Capítulo 1

Introdução

Neste trabalho estamos desenvolvendo um serviço de procura por homólogos através de BLASTp (Basic Local Alignment Search Tool) e PSI-BLAST (Position Specific Iterated Basic Local Alignment Search Tool) integrado a técnicas de aprendizado de máquina e mineração de dados. Sendo que o BLASTp ou PSI-BLAST são as fontes dos atributos (*features*) para os algoritmos de aprendizado de máquina.

1.1 Algoritmos de Aprendizado de Máquina

Aprendizado de máquina é o ramo da inteligência artificial (IA) que estuda e constrói sistemas que podem aprender com os dados. Este ramo da IA também já foi descrito como "O campo de estudo que dá aos computadores a habilidade de aprender sem ser explicitamente programado", Simon (2013). E Mitchel dá uma definição mais completa e formal como "Diz-se que um programa de computador aprende com a experiência E com respeito a uma classe de tarefas T e medida de performance P, se sua performance na tarefa T, mensurada por P, melhora com a experiência E", Mitchell (1997).

Duas formas, principais, de aprendizado de máquina são comumente usadas, supervisionado e não supervisionado. Na primeira os dados são treinados em dados cujas classes são conhecidas e o algoritmo aprende a classificar dados novos nestas classes. Na segunda, o algoritmo encontra regularidades estatísticas nos dados. Um exemplo muito relevante de aprendizado não supervisionado é o PSI-BLAST (Altschul et al. (1997)). Este programa desenvolvido pelo NCBI usa uma técnica chamada PWM (*Position Weighted Matrix*) (Stormo et al. (1982), Xia (2012)) e os pesos desta matriz são recalculados a cada iteração usando os dados que ela encontrou na iteração anterior. Este processo é feito de maneira não supervisionada. Uma revisão mais detalhada sobre PSI-BLAST será feita mais adiante e nesta seção nós nos concentraremos em aprendizado supervisionado.

1.1.1 Rede Neural

Rede neural é uma classe ampla de algoritmos, fracamente, inspiradas no funcionamento do cérebro. Esta classe inclui tanto métodos supervisionados quanto não supervisionados. Neste trabalho nós nos concentraremos em um tipo de rede supervisionada chamada MLP (*Multi Layer Perceptron*) (Collobert and Bengio (2004)).

Antes de introduzir MLP, uma breve introdução sobre regressão logística pode contribuir para a compreensão de MLPs. A regressão logística, também chamada em algumas situações de classificador (Bishop (2006)), é definida pela função 1.1. Esta função pondera os diversos

atributos a ela passados e classifica os dados como positivos ou negativos. Esta ponderação é basicamente linear e esta função não é capaz de mudar o peso que ela dá a um atributo em resposta ao valor de outro atributo, no que poderia ser chamado de "operação estilo XOR".

$$y = \frac{1}{1 + \exp(-f(x))} \tag{1.1}$$

Observe nesta função que dentro do expoente existe um "f(**x**)". Esta função em geral é algum tipo de equação linear com as variáveis (**x**), podendo ser polinomial e ter produto entre atributos (*cross-product*).

A rede neural (MLP), por outro lado, é capaz de fazer predições não lineares, tal como o XOR. Nesta técnica múltiplos classificadores, semelhantes ao descrito anteriormente, são dispostos em camada de maneira que a saída dos classificadores de uma camada se tornam as entradas dos classificadores da próxima camada. Nestas redes a primeira camada de neurônios recebe os dados externos e é chamada de camada de entrada; a ultima camada gera a resposta da rede e é chamada de camada de saída. As camadas existentes entre estas duas, se existirem, são chamadas de camadas ocultas.

Esta arquitetura em camada permite que a rede faça predições não lineares. A rede é capaz de se "lembrar" do valor de uma variável quando avalia o valor de uma outra. De fato, de acordo com Hornik et al. (1989), uma rede neural com pelo menos uma camada oculta é capaz de simular qualquer função matemática.

A capacidade da rede de se "lembrar" dos dados é no entanto uma característica dúbia. Essa memória permite à rede se lembrar de detalhes dos dados de treino que nem sempre se repetem em dados novos, levando a rede ao *overfitting*. Ou seja, a rede tem uma taxa de acerto alta no treino, mas baixa no teste. Esse problema em redes neurais é tanto maior quanto mais neurônios a rede tenha. De fato, a capacidade classificatória da rede pode ser de tal modo afetada por essa "excessiva memória de detalhes irrelevantes" que seu desempenho no teste pode até mesmo cair quando se aumenta a quantidade de neurônios (Tetko et al. (1995)).

Uma família de técnicas muito comuns para reduzir o *overfitting* é a regularização. Duas das maneiras mais comuns são regularização quadrática e interrupção precoce (*early stopping*). Na primeira, dito de maneira bem grosseira e breve, os pesos das sinapses são adicionados à função de erro usada no treinamento. Isso faz com que os pesos cresçam menos. É uma espécie de "suavização" da rede. Na segunda, o resultado do treino a cada iteração é checado em um conjunto de dados externos permitindo que se pare o treinamento quanto o erro nos dados de validação comecem a aumentar (Bishop (2006)). Uma nota sobre esta ultima técnica é que ela previne o que é chamado em alguns casos na literatura de *overtraining* (Tetko et al. (1995)). Este problema consiste no fato de que em algum momento durante o treinamento a rede para de aprender fatos relevantes dos dados e passa a aprender detalhes irrelevantes.

Um outro ponto a se considerar quando se trabalha com rede neural é o treinamento convergir para ótimo local. Este é um problema constante em treinamento não só de redes neurais, mas de algoritmos de aprendizado de máquina e se deve ao fato de que a função de desvio, entre calculado e observado, usada para treinar estes algoritmos, possuem vários pontos de mínimo local, também chamados neste caso de ótimos locais. As técnicas de treinamento tradicionais como gradiente descendente e gradiente conjugado alteram os pesos da rede de

maneira proporcional à derivada da função de desvio. Quando esta função chega ao ponto de mínimo a derivada é zero e a pesquisa converge (Avriel (2003)). Algumas técnicas foram desenvolvidas para se tratar este problema e fazer com que a função de treinamento "suba o morro". Ou seja "aceite" piorar o resultado momentaneamente para tentar melhorar em seguida. Uma destas técnicas é algoritmo genético que é muito aplicado no treinamento de redes neurais (Bornholdt and Graudenz (1992)).

1.1.2 Random Forest

Uma outra técnica de aprendizado de máquina que é importante descrever nesta introdução é *random forest*. Este método é baseado no conceito de árvore de classificação como aquela mostrada na figura 1.1. Como se pode ver nessa figura a árvore seleciona uma variável e testa seu valor como maior ou menor que um limiar e conforme o resultado a árvore decide qual o próximo teste a ser feito. Este processo continua até se chegar a uma das folhas da árvore. Associado às folhas estão as predições que a árvore faz.

A *random forest* é um algoritmo que treina várias árvores, cada uma com uma amostra com reposição dos atributos (*features*) disponíveis e, em seguida, cada árvore dá um voto e a resposta da *random forest* é a média destes votos (Breiman (2001)).

A forma como a *random forest* seleciona os atributos (*features*), treina as árvores e reúne seus resultados é chamado de *bagging* (Breiman (2001), Breiman (1996)). Esta terminologia enfoca a forma como os modelos são construídos com amostragens de atributos. Nós chamaremos de *ensemble* por média para enfocar que o *ensemble* é a média das várias predições.

1.1.3 Ensemble de Modelos

Ensemble de modelos é uma técnica pela qual as predições de diversos modelos, no caso destes trabalhos classificadores, são reunidas para se fazer uma única classificação (Dietterich (2000)). Este método já foi introduzido na seção anterior, uma vez que, como dito, *random forest* é um *ensemble* de árvores de classificação. No caso da *random forest*, tínhamos múltiplas árvores construídas com diferentes subconjuntos de variáveis e o *ensemble* é feito pela média das predições. No entanto, o *ensemble* também pode ser feito entre modelos completamente diferentes tal como rede neural e *random forest*.

No caso da *random forest* o *ensemble* era feito pela média. No entanto, quando se trabalha com modelos diferentes, tal como rede neural e *random forest*, faz sentido formas mais sofisticadas de *ensemble* tal como "*ensemble* linear", onde um modelo linear pondera a importância das predições de cada classificador. Neste caso, o próprio *ensemble* precisa ser treinado em dados que não tenham sido usados para se treinar os modelos anteriores (Silva et al. (2012)).

É claro que perguntas importantes que cabem aqui, quais sejam: qual a vantagem de se fazer todo esse processo? Qual o ganho da reunião de muitos modelos em detrimento de um? Afinal todos eles não fazem as mesmas predições?

Será ilustrativo começar respondendo à última pergunta. Quando se treina diferentes modelos, as respostas que estes geram, variam um pouco, pode se dizer que estas respostas em geral variam em torno das respostas corretas. Ao se fazer *ensemble* de muitos modelos, em geral, a resposta da maioria é a resposta correta. Em outras palavras se todos os modelos retornam

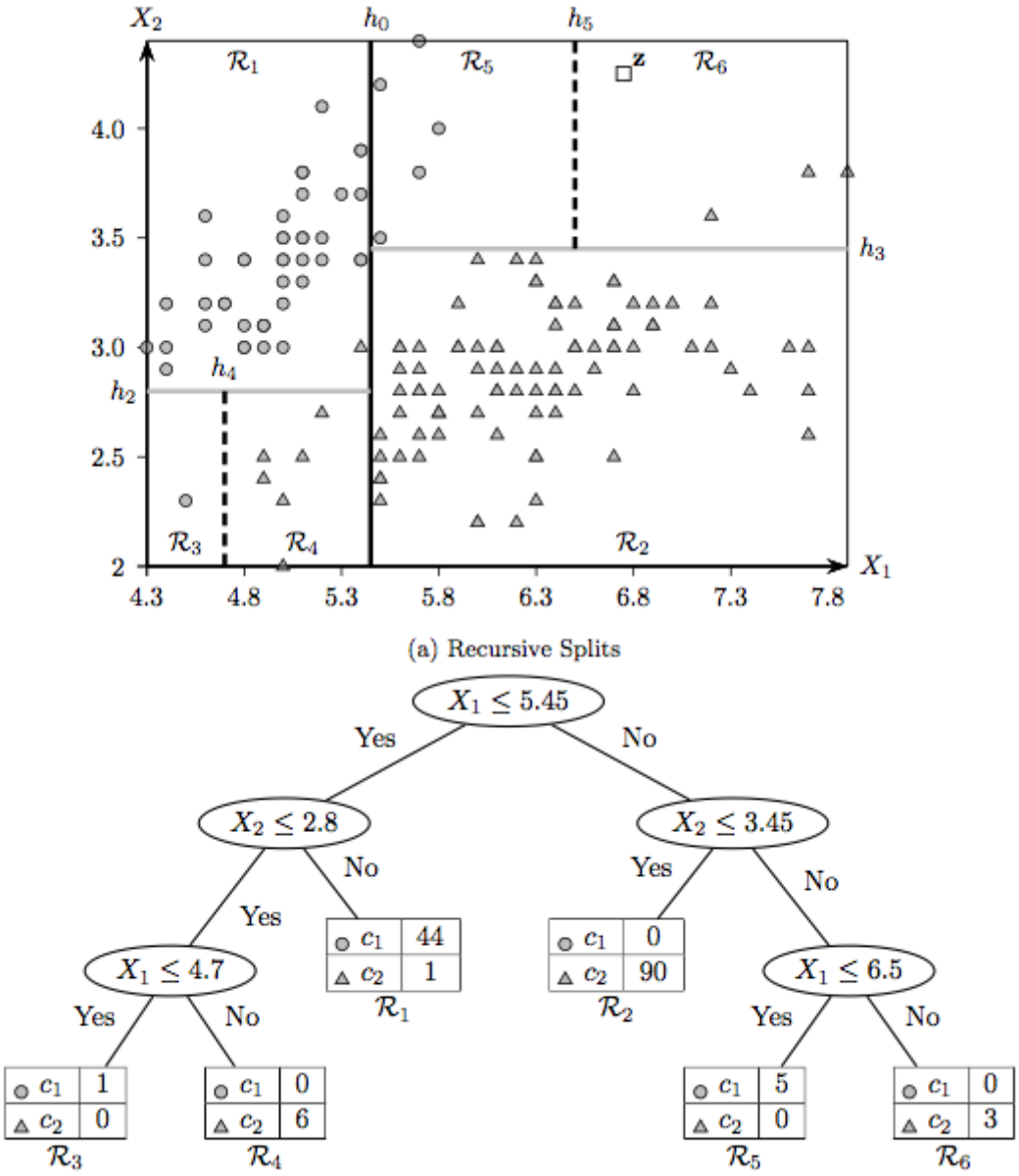

Figura 1.1: Figura mostrando uma árvore de decisão. A figura acima mostra como o espaço foi particionado. A figura abaixo mostra a árvore com o conjunto de decisões que definem o particionamento acima. Observe que nas folhas estão as contagens de objetos em cada classe dentro daquele nó (particionamento). Esta figura foi obtida no livro de Meira and Zaki (2013).

uma resposta "A" e um único modelo discorda dando uma resposta "B" é provável que esta resposta "B" seja um erro daquele modelo em particular (Dietterich (2000), Opitz and Maclin (1999)).

Um outro ganho em se trabalhar com *ensemble*, que decorre da explicação acima, é que esta é uma ótima forma de se lidar com *overfitting*. No caso do *ensemble* de árvores da *random forest* cada árvore é construída de uma maneira propensa a *overfitting*. Em outras palavras, a árvore individual da *random forest* é, propositalmente, um péssimo classificador. No entanto, o *ensemble* de dezenas ou até mesmo centenas de árvores gera um classificador final com baixíssima taxa de *overfitting*. Em estudos realizados por Krogh and Sollich (1996), também ficou demonstrado que o *ensemble* de classificadores lineares com alta taxa de *overfitting* levou a um classificador final robusto. Outro resultado que decorre das características de *ensemble*, tais como as explicadas anteriormente, é que o *ensemble* de modelos gera um predição com menor variância entre treinamentos.

Uma nota final sobre *ensemble* que decorre de nossa experiência pessoal e de alguns dados da literatura é que uma boa regra para se criar um bom *ensemble* de modelos é escolher modelos que tenham desempenho semelhante (por exemplo mesmo AUC (Area Under the Curve) no teste) mas que tenham alta variância entre suas predições (Silva et al. (2012)).

1.1.4 Uma Breve Revisão Sobre o Uso de Modelos de Aprendizagem de Máquina em Bioinformática

Diversos trabalhos na literatura reportam o uso de técnicas de aprendizado de máquina em bioinformática. Nesta revisão não estamos considerando BLAST. Nós nos concentraremos, na verdade, em técnicas como rede neural, *Suport Vector Machine* (SVM), *random forest*, entre outros. Em uma revisão semi direcionada sobre o assunto, encontramos diversos trabalhos que utilizam aprendizado de máquina para atender a variados propósitos como predição de estrutura secundária, predição de localização celular, entre outros.

Nesta revisão, encontramos vários trabalhos que reportam o uso de SVM (Suport Vector Machine) e rede neural para busca e identificação de homólogos distantes. Em um destes trabalhos (Shah et al. (2008)) foi desenvolvida uma ferramenta de busca de homólogos distantes que usa uma técnica semi supervisionada de SVM. Esta ferramenta começa com um treinamento em uma base de dados curada e então novas proteínas são adicionadas. O trabalho de Shameer et al. (2009), por outro lado, combina PSSM (Position Specific Score Matrix) com SVM para atingir um objetivo semelhante ao de Shah et al. (2008). Neste trabalho, redes neurais foram usadas para encontrar a melhor PSSM para representar uma família de proteínas Pfam (Protein family). Esta PSSM foi então usada para fazer busca de homólogos distantes daquela família. Um outro trabalho que utilizou redes neurais para busca de homólogos distantes, que encontramos em nossa revisão, foi Hochreiter et al. (2007). Neste trabalho, uma rede foi treinada para encontrar padrões indicativos de que a proteína pertença a uma determinada classe, bem como, padrões indicativos de que ela não pertença, e baseado nestes padrões, a ferramenta decide. A ferramenta de Liu et al. (2008) combina PSI-BLAST com SVM para o mesmo objetivo. Neste caso, o PSI-BLAST é utilizado para recrutar proteínas a partir das quais o autor extrai uma série de características relevantes, tais como os n-gramas mas frequentes, que são utilizadas para

alimentar uma SVM que detecta homólogos distantes.

Uma outra linha de estudo que utiliza aprendizado de máquina em bioinformática, de acordo com a revisão aqui apresentada, é a predição de "função" de proteína. Mais especificamente, alguns dos trabalhos que encontramos, reportaram a criação de ferramentas que fazem predições muito especificas, como Garg and Gupta (2008), que prediz o fator de virulência de uma proteína de bacteriana, utilizando SVM. Em outro trabalho (Kumar et al. (2011)) foi desenvolvido um modelo capaz de predizer se uma lectina é uma carcinolectina ou não. Para isso a ferramenta baseada em SVM utiliza composição de dipeptídeos e perfis de PSSM de PSI-BLAST. Também encontramos um trabalho semelhante (Kalita and at al (2008)), que usa composição de dipeptídeos e perfis de PSSM de PSI-BLAST, juntamente com a composição de estrutura secundária, para alimentar uma SVM que prediz se uma sequência é uma ciclina.

A determinação de localização de proteínas é um outro tópico de interesse em biologia e várias ferramentas baseadas em técnicas de aprendizado de máquina têm sido propostas para esta tarefa. A ferramenta de Rashid et al. (2007), por exemplo, utiliza perfis de PSSM de PSI-BLAST para alimentar uma SVM que prediz localização sub-celular de proteínas de micobactérias. O modelo de Zou et al. (2007) também faz predição de localização celular de proteínas de eucariotos, no entanto, esta ferramenta usa rede neural e PSI-BLAST para esta tarefa. A ferramenta de Oua et al. (2008) identifica beta-barril de proteínas de membrana utilizando perfis de PSSMs criadas por PSI-BLAST e redes neurais RBF (*Radial Based Function*).

Talvez um dos campos mais férteis para o uso de técnicas de aprendizado de máquina, mormente técnicas do tipo "PSSM-PSI-BLAST-ML" (PSSM-PSI-BLAST-Machine Learn), seja a predição de estrutura secundária. Um exemplo disso pode ser encontrado no trabalho de Chatterjee et al. (2011) que utilizou PSSMs geradas por PSI-BLAST juntamente com características físico-químicas dos aminoácidos para alimentar um modelo de SVM que prediz estrutura secundária de proteínas. Por outro lado, o trabalho de Ghanty and at al (2013) não utilizou a técnica 'PSSM-ML', ao invés disso ele calculou as probabilidades posição específica e posição inespecífica de ocorrência de estrutura secundária e treinou um modelo híbrido de rede neural e SVM com cada um.

Próximo da linha de raciocínio de estrutura secundária estão os estudos sobre enovelamento, como o trabalho de Yan et al. (2009) que desenvolveu a ferramenta DescFold que usa SVM para prever enovelamento de proteínas ou a ferramenta SVM-fold (Melvin and at al (2007)) que classifica proteínas em classes estruturais (desnecessário dizer que ela usa SVM).

Também próximo desta linha de pensamento dos dois parágrafos anteriores está a ferramenta PPRODO (Sim et al. (2005)) que usa perfis de PSSM de PSI-BLAST e rede neural para identificar limites de domínios de proteínas.

Em nossa busca por trabalhos na área de bioinformática e aprendizado de máquina encontramos alguns trabalhos relacionados à predição de resíduos de ligação a DNA e RNA. Entre estes está o trabalho de Wang et al. (2010) que treinaram uma SVM para predição de resíduos de aminoácidos que se ligam a RNA e DNA. Uma ferramenta semelhante foi proposta por Kumar and at al (2011) em que a predição de resíduos de ligação a DNA foi feita por uma SVM que utiliza composição de dipeptídeos e perfis de PSSM de PSI-BLAST. Por fim, no trabalho de Chauhan et al. (2009) um modelo de SVM baseado em sequência primária de proteína e outro

de SVM baseado em perfil de PSSM de PSI-BLAST foram construídos para predizer resíduos de ligação a ATP.

Como se pode ver nesta breve revisão, o uso de perfis de PSSM alimentando algoritmos de aprendizado de máquina é uma técnica popular em bioinformática. Também se pode ver que o uso destes algoritmos é bem frequente, uma vez que, pudemos encontrar muitos trabalhos utilizando SVM e rede neural em uma busca semi direcionada na literatura.

1.2 Mineração de Texto

Mineração de texto diz respeito a técnicas para se extrair informação de textos. O tema é bastante amplo e envolve áreas como classificação de texto (por exemplo e-mails com SPAM e não-SPAM), correção automática de resposta a pergunta, recuperação de textos relevantes dado uma *query*, entre outros.

Um procedimento frequente em mineração de texto, que em geral é essencial para o bom desempenho de qualquer algoritmo, é a realização de alguns pré-processamentos. Como exemplo dos mais comuns temos a remoção de caracteres especiais e pontuação, remoção de número em alguns casos, remoção de *stop-words* (mais sobre isso a seguir), e *stemming*.

Duas das técnicas citadas acima que merecem maior atenção são remoção de *stop-words* e *stemming*. A primeira consiste em remover do texto palavras pouco informativas, como termos de ligação (”para”, ”na”, ”a”, ”uma”, etc). Estas palavras podem ser encontradas em praticamente qualquer texto e não contribuem para sua compreensão ou classificação. Existem listas destas palavras que podem ser encontradas livremente na internet, como por exemplo a lista de *stop-words* do MySQL (Wilbur and Sirotkin (1992)).

A segunda técnica citada chamada *stemming* consiste em tentar extrair o radical das palavras, de forma que por exemplo ”muitos” e ”muita” se tornem ”muit”. Várias bibliotecas estão disponíveis livremente na internet. A mais famosa, provavelmente, é PorterStem (Paice (1996), Willett (2006)).

1.2.1 Representação como bag of words

Nesta introdução iremos focar em mineração de dados baseada na representação de texto como *bag of words*. Esta técnica, como a usamos neste trabalho, consiste em representar o texto como um vetor binário onde cada posição do vetor corresponde a uma palavra do vocabulário e esta posição pode estar preenchida com 0 caso a palavra não tenha ocorrido no texto ou 1 caso a palavra tenha ocorrido pelo menos uma vez.

A representação como *bag of words* é na verdade apenas o início do processo. Uma vez que se tenha uma coleção destes vetores eles podem ser usados em quaisquer dos métodos de aprendizado de máquina tradicionais. Por exemplo, esses vetores podem ser usados como atributos (*features*) para algoritmos de classificação como rede neural, *random forest*, SVM, ou rede bayesiana. Podem ser agrupados com algoritmos como *K-means* ou kNN (*k Nearest Neighbor*).

Esta representação vetorial pode tirar proveito de técnicas de álgebra linear. Métodos comuns encontrados na literatura para se comparar textos, representados da maneira descrita, são distância euclidiana e cosseno do ângulo entre dois vetores. É comum também o uso de

técnicas de fatorização de matriz. As mais usadas são SVD (Singular Value Decomposition) e NNF (Non Negative Factorization). Estas visam a redução da dimensionalidade do problema através da eliminação de variáveis (*features*) pouco informativas (Klema and Laub (1980), Catral and at al. (2004)).

Ao se tratar os textos desta forma grande parte da informação contida na semântica é perdida, no entanto, esta técnica tem a vantagem de ser simples e, em muitas aplicações, tem se mostrado eficaz. De fato pode-se recuperar muita informação em um texto apenas olhando as palavras que este contem. Não obstante, é possível se recuperar parte da semântica usando os chamados *n-grams*. Nesta técnica não apenas se representam as palavras mas também pares de palavras (*di-grams*), ou trios (*tri-grams*) e assim por diante (Tomovica et al. (2006), Mason et al. (2009)).

1.3 Uma breve introdução sobre busca de homólogos

A busca por homólogos é de grande interesse em biologia. No campo da bioinformática vários programas e algoritmos foram criados para buscar e medir homologias entre sequências de aminoácidos e nucleotídeos. A estratégia mais comumente utilizada para se alcançar este objetivo é o alinhamento com algoritmos como Smith-Waterman (Smith and Waterman (1981)) e BLAST (Altschul (1990)). No entanto estes métodos, embora poderosos, falham em reconhecer sequências com homologias distantes.

Para se detectar homologias distantes varias estratégias também foram desenvolvidas. A mais comum compara uma sequência com o perfil de uma família, como o PSI-BLAST (Altschul et al. (1997)) e Profiles (Gribskov and at al (1987)). Por outro lado programas como BLOCKS (Henikoff and Henikoff (1991)) e FFAS (Rychlewski and at al (2000)) comparam dois perfis entre si. Uma outra metodologia, igualmente popular, são as cadeias de Markov ocultas (citehmm e Johnson et al. (2010)). Uma quarta estratégia consiste em usar a transitividade da homologia. Ou seja, se ”I” é homólogo à *query* ”Q” e ”T” é homólogo ”I” então a homologia entre ”Q” e ”T” pode ser estabelecida (Salamov and at al (1999), Weizhong and at al (2000)). Por fim muitas ferramentas usam informação sobre estrutura secundária e ou terciária para detectar homologias distantes (Moult (1999)).

1.4 Análise de Sequências Proteicas

1.4.1 Alinhadores

Os métodos de alinhamento de sequência podem ser divididos entre alinhamento global e alinhamento local. No global é feita uma tentativa de se alinhar toda a sequência, enquanto no local são alinhadas subsequências, ou regiões.

Os alinhadores globais são considerados a melhor forma de analisar relações evolutivas entre sequências. Sendo que o mais conhecido é o de Needleman-Wunsch (Needleman and Wunsch (1970)). Existem ferramentas que fazem alinhamentos globais e múltiplos para facilitar essas análises evolutivas como o CLUSTAL (Larkin and et al (2007)).

O alinhamento local, por outro lado, é bastante usado para a anotação funcional, pois é capaz de detectar regiões conservadas (domínios conservados) entre sequências. Os principais métodos para alinhamento local, em bioinformática, são o algoritmo de Smith-Waterman (Smith and Waterman (1981)) e o algoritmo BLAST (Altschul (1990)).

Além da divisão global/local, outra forma independente de classificar os alinhadores é entre par a par e múltiplo. Nos primeiros, como o BLAST, as sequências são alinhadas duas a duas. No caso do BLAST, uma *query* é alinhada com um *subject*, depois com outro e assim por diante. Já os alinhadores múltiplos, como o CLUSTAL, alinham múltiplas sequências ao mesmo tempo.

1.4.2 BLASTp e as Matrizes BLOSUM

O BLASTp é um algoritmo de alinhamento local publicado em 1991. Este programa trabalha com alinhamento local e matrizes de pontuação. Os *match/mismatch* dos alinhamentos são pontuados conforme o valor que a matriz atribui para aquela substituição de aminoácidos. As matrizes mais comumente usadas são PAM (Point Accepted Mutation) (Dayhoff (1978)) e as BLOSUMs (Henikoff and Henikoff (1992)). O BLAST também pontua (negativamente) abertura e extensão de gaps. Estas penalidades também estão definidas no arquivo das matrizes.

Embora o BLASTp possa usar outras matrizes de pontuação como PAM as BLOSUM são as mais utilizadas e a BLOSUM62 é a matriz padrão. Estas matrizes foram construídas medindo-se as frequências de substituição de aminoácidos entre proteínas e comparando-se com o esperado por mero acaso. Em outras palavras, a frequência de *match/mismatch* foi medida em alinhamentos de proteínas (Henikoff and Henikoff (1992), Fabris et al. (2007)). A equação 1.2 mostra isso:

$$s = \log_2 \frac{fa12}{fa1 * fa2} \tag{1.2}$$

Onde "fa1" é a frequência do aminoácido "a1" no banco de dados, "fa2" a frequência do aminoácido "a2", "fa12" a frequência de troca do aminoácido "a1" pelo aminoácido "a2" em um conjunto de alinhamentos e a multiplicação "fa1*fa2" é a frequência de troca esperada por mero acaso. O valor "s" é a pontuação atribuída a esta substituição. Esta pontuação, por razões de desempenho computacional, é arredondada para o inteiro mais próximo. A característica importante desta pontuação é que as substituições que ocorrem acima do esperado por mero acaso são pontuadas positivamente; aquelas que ocorrem menos que o esperado por, mero acaso, são pontuadas negativamente e aquelas que ocorrem com frequência igual a esperada recebem pontuação zero (Henikoff and Henikoff (1992), Fabris et al. (2007)).

A equação 1.2 foi aplicada a cada uma das 210 substituições possíveis (A por A, A por C, A por D ...). E, assim, as matrizes BLOSUM foram criadas. Para se medir as frequências das substituições "fa12" foram feitos alinhamentos na base de dados BLOCKS. Esta base contém um amplo espectro de proteínas e as medidas feitas nesta base representam o universo e não uma família ou grupo de proteínas em particular. No entanto, poderia haver um viés ligado ao fato de que certos grupos de proteínas possuem muitos membros e outros possuem poucos membros. Para minorar este viés, proteínas com identidade maior que um certo limiar foram agrupadas e uma única medida foi feita para todo o grupo. Grosso modo o grupo inteiro foi

tratado como uma única proteína. O limiar descrito anteriormente define as diversas BLOSUMs. Na BLOSUM62 o limiar é 62% de identidade, na BLOSUM80 80% de identidade e assim por diante (Henikoff and Henikoff (1992), Henikoff and Henikoff (1991)).

A partir do que foi exposto acima, pode-se ver que alinhamentos pontuados com a BLOSUM tendem a recuperar proteínas com aminoácidos semelhantes aos da *query*. Em outras palavras a BLOSUM satisfaz a seguinte intuição: "quais aminoácidos 'x' poderiam ser substituídos por quais aminoácido 'z' de maneira que a sequência permanecesse semelhante". Isso vai além da distância edicional comum em muitos alinhadores.

Uma caraterística interessante da BLOSUM é que ela não parte de teorias preconcebidas sobre qual aminoácido se parece com qual (quimicamente ou etc), ao invés disso, os criadores da matriz permitiram que ela aprendesse com os dados.

O BLASTp além de pontuar seus alinhamentos com a BLOSUM também possui uma estatística adicional extremamente relevante. Estamos falando, é claro, do conceito de *E-value*. Este valor mede quantos alinhamentos com pontuação 's' se espera encontrar por mero acaso em um banco de dados com tamanho "nxm". Onde 'm' e o tamanho do banco multi-fasta e "n" é o tamanho da *query*. Em outras palavras, este valor leva em conta que à medida que os bancos de dados de proteínas aumentam, aumenta também a probabilidade de que proteínas não relacionadas à *query* passem a ter uma certa pontuação. Igualmente, uma proteína grande tende a ter pontuação mais alta, pois a pontuação será a soma de uma série maior de pontuações de *match/mismatches*. E é claro que uma proteína muito pouco semelhante à *query* pode acabar recebendo uma pontuação relativamente "alta". O *E-value* pondera todos estes fatos e um *subject* não recebe *E-value* baixo se sua pontuação for fruto de um acaso causado por sequência de aminoácidos e banco de dados grandes (Altschul (1990), Karlin and Altschul (1990)).

1.4.3 PSI-BLAST

A BLOSUM possui grandes virtudes, no entanto, possui também fraquezas. A primeira destas fraquezas é o fato de que ela é posição inespecífica, ou seja, ela não leva em conta a posição do aminoácido na proteína. Por exemplo, em enzimas, a existência de alguns aminoácidos específicos em uma posição particular da proteína é essencial para sua função. A BLOSUM e o BLASTp não consideram isso. Outra fraqueza é o fato dela ser genérica e em muitos casos uma substituição de aminoácidos que, no geral, é frequente pode ser rara em uma família de proteínas em particular. Na seção anterior apresentamos a inespecificidade da BLOSUM como virtude e agora estamos apresentando como fraqueza.

A resposta para estes problemas apresentada pelo NCBI foi o PSI-BLAST (*Position Especific Iterated Basic Local Alignment Search Tool*) (Altschul et al. (1997)). Esta ferramenta trabalha com muitas das ideias apresentadas anteriormente. De fato, o PSI-BLAST inicia-se como um BLASTp. A partir dos resultados encontrados pelo BLASTp uma nova matriz de pontuações é construída. Esta matriz é calculada com a equação 1.2 usando as frequências de substituições encontradas entre as proteínas recrutadas. Uma nova iteração é feita usando a matriz recalculada e as novas proteínas recrutadas são usadas para se calcular uma nova matriz. Este processo se repete até que os resultados encontrados na iteração "i" sejam os mesmos encontrados na iteração "i-1" (convergência), ou até o número máximo de iterações definido pelo usuário seja

alcançado. Esta técnica não supervisionada de recálculo iterativo da matriz de pontuação permite ao PSI-BLAST alcançar a especificidade que o BLASTp não possui (Altschul et al. (1997), Schäffer and et al (2000)). Embora o PSI-BLAST possa prosseguir por tantas iterações quanto o usuário desejar, de acordo com Schäffer and et al (2000) e com Lee et al. (2008) a maior parte dos *matches* que podem ser localizados são encontrados com 5 ou 6 iterações. Sendo que esta é uma das principais razões porque neste trabalho nós limitamos a pesquisa PSI-BLAST a 5 iterações.

No parágrafo anterior dissemos que a matriz é recalculada usando as proteínas encontradas na iteração anterior. No entanto, um fato importante foi omitido. Diferente da BLOSUM que é uma matriz em $\mathbb{R}^{20x20}$, as matrizes que o PSI-BLAST calcular estão em $\mathbb{R}^{20xm}$, onde "m" é o tamanho da *query*. Ou seja os valores na matriz não são de substituição do aminoácido "i" pelo aminoácido "j", mas sim do aminoácido "i" pelo aminoácido "j" na posição "k". Isso permite ao PSI-BLAST considerar aminoácidos posição específicos importantes em uma família de proteínas como os aminoácidos catalíticos de uma enzima (Altschul et al. (1997)).

Um problema relacionado à maneira não supervisionada que o PSI-BLAST usa para calcular os pesos da matriz é que ele eventualmente inclui sequências não relacionadas o que acaba corrompendo a matriz (Jones and Swindells (2002)).

O trabalho de Lee et al. (2008) trata do problema de corrupção de matriz e desenvolve um método para solução deste problema. A estratégia utilizada pelos autores combina os resultados da primeira iteração, onde a matriz foi menos corrompida, com os da última iteração que tem maior recrutamento. A partir destes resultados os autores criaram uma forma de ranquear os resultados superior ao *e-value*. A estratégia usada pelos autores lembra em alto nível o que será apresentado neste trabalho.

Não obstante os problemas de corrupção da matriz, o PSI-BLAST é considerado estado da arte em termos de buscador de homólogos distantes (Mullera et al. (1999)). Várias ferramentas desenvolvidas para este tipo de tarefa usam PSI-BLAST ou comparam seus resultados com PSI-BLAST.

Ferramentas Baseadas em PSI-BLAST

Em uma breve busca na literatura sobre ferramentas baseadas em PSI-BLAST, encontramos vários trabalhos sobre busca de homólogos distantes e predição de estrutura secundária. Exemplo de uma dessas ferramentas inclui o IMPALA (Schaffer and at al (1999)) que faz uma pesquisa em um banco de dados de PSSMs geradas por PSI-BLAST. O autor argumenta em favor do PSI-BLAST que a busca com PSSM se provou efetiva em encontrar homólogos distantes. Uma outra destas ferramentas é o Re-searcher descrita por Repsys et al. (2008) que, periodicamente, dispara PSI-BLAST para detectar de maneira recorrente novos homólogos para uma proteína *query*. No trabalho de Anand et al. (2005) o autor demonstra que o uso de múltiplas PSSMs, oriundas de PSI-BLAST, é mais efetiva que uma única PSSM bem como de HMM. O método descrito por Uehara et al. (2004) filtra resultados de PSI-BLAST usando informações de estrutura secundária predita por uma rede neural.

Um outro campo em que se emprega comumente o PSI-BLAST é na criação de perfis de PSSM que são usado para predição de estrutura secundária ou de classe estrutural da proteína. Vários destes trabalhos foram vistos na seção 1.1.4. Citaremos mais dois trabalhos deste tipo

aqui. O primeiro deles é o trabalho de Liu et al. (2012) que descreve um método de extração de características da PSSM de PSI-BLAST para predição de classe estrutural. Já a ferramenta SPARROW, descrita por Bettella et al. (2012) prediz estrutura secundária de proteínas através de um perfil de sequências obtidas com PSI-BLAST.

1.4.4 Alinhamentos Múltiplos e Árvores Filogenéticas

Os alinhadores múltiplos são uma extensão dos alinhadores par a par, sendo que nesta técnica várias sequências são alinhadas. Estes métodos são usados para achar motivos conservados em proteínas, construir árvores e calcular distância filogenética, entre outros. Várias ferramentas de alinhamento múltiplo foram descritas e estão disponíveis na literatura, como o CLUSTAL (Larkin and et al (2007)), que é provavelmente o mais famoso, e o MUSCLE (Edgar (2004)).

Como mencionado anteriormente, alinhamentos múltiplos são usados para se construir árvores filogenéticas. A maneira padrão de se construir estas árvores é primeiro fazer o alinhamento que é, então, carregado nos programas de geração de árvore. Assim como os programas de alinhamentos múltiplos, várias ferramentas de geração de árvore foram criadas, como o MEGA (Tamura et al. (2007)), o FastTree e o FastTree2 (Price et al. (2010)). Este último traz melhorias em relação ao anterior que o deixa mais rápido. Também é importante citar o PHYLIP (PHYLogeny Inference Package) que pode ser usado via web ou baixado gratuitamente. Este pacote inclui vários métodos de criação de árvores como parcimônia, matriz de distâncias, máxima verossimilhança, *bootstrap* e árvore consenso(Felsenstein (2002)).

1.4.5 Modelo de Cadeias de Markov Ocultas

Nas seções anteriores, nós nos concentramos em comparação de sequência através de alinhamento e pontuação com técnicas de PWM (Position Weight Matrix), como BLAST. Agora, iremos mostrar uma outra forma de análise completamente diferente, mas igualmente poderosa: as cadeias de Markov ocultas. Antes de falar sobre as cadeias ocultas, uma breve explicação sobre cadeia de Markov se faz necessária. Esta técnica é baseada na equação 1.3

$$P(Xn+1 = j|X1 = i0, X2 = i1, \ldots, Xn = in) = (P)inj \tag{1.3}$$

A equação acima diz que a probabilidade do próximo valor da série ser "x" (Xn+1 = x) é condicionada pelos "n" valores anteriores. O valor exato de "n" determina o que é chamado de ordem da cadeia. Por exemplo, na cadeia de ordem 1 existe a probabilidade do próximo valor ser "x" dado que o valor anterior foi "x1"; na cadeia de ordem 2 a probabilidade do próximo valor ser "x" é condicionada pelos dois valores anteriores e assim por diante. No cálculo de cadeias de Markov, uma matriz, chamada matriz de transição, é calculada com as probabilidades de todos os valores possíveis, dadas todas as combinações de "n" valores anteriores (Stroock (2005)).

Nas cadeias de Markov simples como descrito acima, as probabilidades de transição podem ser observadas e medidas diretamente. Nas cadeias de Markov ocultas (HMM ou Hidden Markov Model), por outro lado, estas probabilidades não podem ser medidas e apenas os

resultados que dependem destas probabilidades podem ser vistos (Krogh et al. (1994)).

Juntamente com PSI-BLAST, HMMs são consideradas a maneira padrão para busca de homólogos distantes. Existe uma miríade de ferramentas que fazem uso destas técnicas, como o trabalho de Johnson et al. (2010) que argumenta que HMM é uma das principais maneiras de se buscar homólogos distantes. Nesse trabalho, os autores desenvolvem uma heurística para os métodos de pontuação.

1.4.6 Estrutura Secundária e Segment Overlap

Nas seções anteriores, focamos em métodos que tratam da análise e comparação da estrutura primária das proteínas, a sequência de aminoácidos. No entanto, estas sequências se enovelam formando estrutura tridimensionais e motivos "intermediários" entre esta estrutura 3D e a estrutura linear 1D. Esse motivos são chamados de estrutura secundária. O conhecimento dessa estrutura ajuda na determinação da estrutura terciária. Além do mais, comparar proteínas por sua estrutura secundária pode revelar homologia que não era evidente ao nível de sequência de aminoácidos.

Vários métodos físico-químicos e computacionais foram propostos para medir ou predizer estruturas secundárias. O dicroísmo circular, por exemplo, é usado para estimar a porcentagem de alfa-hélice e folha beta. Os métodos de difração com raios-X e ressonância magnética nuclear, que revelam a estrutura terciária de uma proteína, também revelam sua estrutura secundária. E estas estruturas secundárias, embebidas dentro da estrutura terciária, podem ser recuperadas pelo algoritmo DSSP (Define Secondary Structure of Proteins) (Kabsch and Sander (1983)). Por outro lado, métodos como Chatterjee et al. (2011) e Bettella et al. (2012), entre outros, são usados para predizer se um aminoácido faz parte de uma alfa-hélice, folha-beta ou *random-coil*. Essas predições ajudam a expandir o conhecimento contido em proteínas, cuja estrutura, foi determinada, experimentalmente, para proteínas homólogas.

Para se medir a qualidade de predições como as dos métodos citados, Rost et al. (1994) e Zemla et al. (1999) propuseram a métrica SOV (Segment OVerlap) que compara a estrutura secundária predita com a estrutura real. O SOV mede a porcentagem de seguimentos de estrutura que se sobrepõem e retorna um número entre 0 e 100%, onde 100% significa que a predição de estrutura, medida em uma proteína conhecida, foi perfeita, e 0% seria predição completamente errada. Os autores argumentam, no entanto, que o SOV entre um par de proteínas aleatórias gira em torno de 37%.

Embora esta técnica tenha sido desenvolvida primariamente para avaliar qualidade de algoritmos de predição, alguns trabalhos desenvolvidos por nosso grupo de pesquisa estão propondo o uso desta técnica para se comparar estrutura de duas proteínas conhecidas.

1.5 Bancos de Dados de Proteínas

Os bancos de dados de proteínas são bases que mantêm informações sobre proteínas, como sua sequência, função, organismos de origem. Muitas destas bases agrupam as sequências em *clusteres* ou famílias. Estes agrupamentos ajudam na anotação de proteínas.

1.5.1 UniProt

O UniProt é um consórcio de banco de dados formado em 2002 pela reunião dos bancos trEMBL, Swiss-Prot, EBI e PIR. A missão primária do UniProt era criar uma base de dados rica e de alta qualidade, com proteínas precisamente anotadas (Apweiler and et al (2004)).

O consórcio pode ser dividido, grosseiramente, em três bases de dados. A primeira dessas é UniRef, que agrupa entradas que compartilham 50%, 90% ou 100% de identidade em uma sequência não redundante. A segunda é o UniParc, que agrupa uma coleção de sequências não redundantes por juntar toda a informação de sequências públicas. Por fim, temos o UniProtKB (UniProt Knowledgebase) que une informações das bases Swiss-Prot (anotação e curadoria manual) e trEMBL (anotação automatizada) (Consortium (2009)). Esta última base continha mais de 23 milhões de sequências em janeiro de 2013 e foi extensamente utilizada neste trabalho.

1.5.2 Kegg

O Kegg é uma base de dados que foi idealizada em 1995 (Kanehisa (1997)) e vem sendo desenvolvida e ampliada desde então. Algumas destas ampliações foram as bases '*Genome*' em 2000 e '*Reaction*' em 2001. Mas a mais importante destas ampliações foi feita em 2002 com a criação do grupo de ortólogos KO (*Kegg-Orthology*) (Kanehisa and et al (2004)).

A base de dados KO agrupa as proteínas em grupos de ortólogos criados por curadoria manual e análise computacional, sendo uma base importante para anotação de proteínas.

1.5.3 Pfam

O PFam é um conjunto de alinhamentos múltiplos de proteínas e perfis de cadeia de Markov oculta. Esta base agrupa as proteínas em famílias e continha 14831 famílias em Novembro de 2013. Esta base inclui uma anotação para cada família na forma de uma descrição textual e faz parte do consórcio InterPro (Bateman and et al (2002)).

1.5.4 PANTHER

O PANTHER é uma base de dados de agrupamentos de proteínas. Esta base foi feita inicialmente com 40 genomas completos mas este universo de dados foi expandido para 82 a partir da versão 8.0 (Mi and at al (2005), Mi et al. (2007), Mi and at al (2010)). A base conta com 6599 *clusteres* sendo que cada um destes possui um perfil de cadeia de Markov oculta. Estes perfis podem ser usados para classificação automática de novas proteínas. O PANTHER possui perfis não só para as famílias (*clusteres*), mas também para subfamílias funcionalmente distintas. Isso assegura a correta classificação funcional de novas proteínas (D.Thomas and at al (2003)).

1.5.5 PDB

O PDB é um banco que agrupa proteínas com estrutura terciária conhecida. Este banco contém mais de 40 mil proteínas com estrutura tridimensional resolvida por difração com raio-X ou ressonância magnética nuclear (Berman (2008)).

As estruturas terciárias contém informações importantes sobre as proteínas e que não podem ser extraídas da sequência de aminoácidos. De fato muitas proteínas que são bastante distintas em sua sequência primária podem ser semelhantes em termos de estrutura terciária.

Capítulo 2

Objetivos

Objetivo Geral

Criar um modelo de análise de sequências capaz de fazer busca por homólogos distantes.

Objetivos Específicos

Aplicar técnicas de aprendizado de máquina supervisionado a resultados de PSI-BLAST de maneira a tirar proveito da alta sensibilidade do PSI-BLAST e da capacidade de discriminação entre positivo e negativo dos classificadores.

Aplicar estes modelos a proteínas de função desconhecida de maneira a sugerir anotação para o máximo delas.

Criar uma ferramenta que implemente estes modelos.

Capítulo 3

Materiais e Métodos

3.1 Recursos computacionais

3.1.1 Hardware e Sistema Operacional

Para realização deste trabalho foram utilizados os recursos computacionais do laboratório Biodados do ICB-UFMG. Esses recursos incluem computadores quad-cores com 4-8 GB de memória RAM e conexão com internet de 100MB/s. Também foi utilizada a máquina pinguim da Faculdade de Medicina de Ribeirão Preto da USP que possui 24 cores e 64 GB de memória RAM e um TB de HD. Todos esses computadores tinham sistema operacional Linux distribuição Red-Hat.

3.1.2 MySQL

Ao longo deste trabalho foram utilizados cliente e servidor de MySQL versão 5 ou superior que pode ser obtida no endereço `http://mysql.com`. Também foi utilizado neste trabalho o componente para comunicação de MySQL com Java chamado *Connector/J* disponível no mesmo endereço.

3.1.3 Java

Para a realização de diversas etapas do presente trabalho foi utilizada a máquina virtual Java versão 6 que pode ser encontrada no endereço `http://java.com`. Para desenvolvimento dos códigos Java foi utilizado o ambiente de desenvolvimento integrado NetBeans versão 7.0 que pode ser encontrado no site `http://netbeans.org`.

3.1.4 Pacote Estatístico R

Neste trabalho foi utilizado o pacote estatístico R versão 3.0.0 que pode ser obtido no endereço `http://r-project.org`. Também foram usados os pacotes de R *randomForest*, *nnet* e *verification*, sendo que este último pacote pode ser instalado gratuitamente no R, através da função *install.package*. Utilizou-se também o ambiente de desenvolvimento R-Studio versão 0.97 que pode ser encontrado em `http://rstudio.com`.

3.1.5 BLAST

Ao longo do trabalho, foi utilizado o pacote BLAST+ versão 2.2.26 desenvolvido pelo NCBI e que pode ser baixado gratuitamente, para propósitos acadêmicos, no endereço `http://blast.ncbi.nlm.nih.gov`. Em particular, foram utilizados os programas *psiblast*, *blastp*, *makeblastdb*, *blastdbcmd* e *blastdb_aliastool*.

3.1.6 MUSCLE e FastTree

Os programas MUSCLE e *FastTree* versão 2.1.7 foram usados neste trabalho para se calcular distâncias filogenéticas e fazer árvores. Estes programas podem ser encontrados e baixados livremente nos sites `http://ebi.ac.uk` e `http://microbesonline.org/fasttree` respectivamente.

3.1.7 MATLAB

O software MATLAB foi utilizado para se calcular SOV. Este programa pode ser encontrado em `http://mathwords.com`, no entanto não é um software gratuito e neste trabalho foi utilizada a licença da universidade.

3.1.8 Outros Recursos Computacionais

Além dos recursos já citados, foi utilizado, neste trabalho, o servidor de PHP versão 5.1.6., bem como recursos presentes nos sistemas operacionais Linux.

3.2 Bases de dados Locais

3.2.1 UniProt

Neste trabalho, foram utilizadas as bases de dados muti-fasta *UniProt* versões de fevereiro de 2012 e janeiro de 2013. Estas bases foram baixadas no site `http://uniprot.org` e utilizadas para se fazer BLAST localmente. Também foi baixado no mesmo site a tabela *idmapping* que foi guardada como uma tabela MySQL. Esta tabela contém relacionamento dos ids das proteínas da base UniProt com outras informações relativas à proteína como taxid, KO, GI, entre outros.

A partir da tabela *idmapping* também foram extraídas as informações relativas ao número de catálogo enzimático *EC-number*.

3.2.2 Tabelas TaxSimple e TaxNames

Neste trabalho foram utilizadas as tabelas *taxSimple* e *taxNames*. A primeira delas contém os clados taxonômicos de cada Organismo. Já a segunda mapeia identificador de táxon (txid) para nome de táxon. Estas tabelas podem ser encontradas no site `http://biodados.icb.ufmg.br` e foram usadas para se fazer LCA (Lowest Common Ancestor). Esta técnica será mais bem descrita mais adiante neste trabalho.

3.2.3 Base de dados PANTHER

A base de dados PANTHER versão de fevereiro de 2012 contendo 40 genomas completos e 6599 *clusteres* de proteínas foi baixada no site `http://pantherdb.org`. Foi baixado o multi-fasta do PANTHER bem como a tabela relacionando cada *protein-id* de PANTHER com seu respectivo *cluster*.

3.2.4 Banco de Dados PANTHER-UniProt

A base de dados PANTHER (3.2.3) e a base de dados UniProt versão de fevereiro de 2012 foram reunidas para formar a base PANTHER-UniProt. Para se fazer esta reunião, primeiro as proteínas PANTHER presentes no UniProt foram descartadas da base PANTHER. Em seguida, esta base de dados PANTHER editada e a base de dados UniProt foram formatadas e reunidas através das ferramentas *makeblastdb* e *blastdb_aliastool* (ver seção 3.3.1).

3.2.5 KEGG

Os genomas dos microrganismos *Escherichia coli K12 DH10B*, *Mycobacterium tuberculosis CDC1551*, *Bacillus subtilis 168* e *Halobacterium sp.* foram obtido a partir da base de dados KEGG em outubro de 2010. Também foi obtido a partir desta base de dados o KO versão de fevereiro de 2012. O endereço do KEGG é `http://genome.jp/kegg/`.

3.3 Execução de SoftWares

3.3.1 BLAST

Neste trabalho, várias sequências de proteínas foram submetidas a BLASTp e PSI-BLAST. Estas submissões foram feitas com as seguintes linhas de comando:

```
blastp -query input -db multi_fasta -evalue 1e-10 \
-comp_based_stats 0 -outfmt 7 -num_alignments 100000 -out output
```

```
psiblast -query input -db multi_fasta -evalue 1e-10 \
-inclusion_ethresh 1e-10 -num_iterations 5 -comp_based_stats 0 \
-outfmt 7 -num_alignments 100000 -out output
```

Onde "*input*" representa um arquivo fasta contendo uma sequência de aminoácidos, "*multi_fasta*" representa um arquivo multi-fasta formatado (ver a seguir sobre formatação) e "*output*" representa o nome do arquivo que se escolhe para salvar a saída do BLAST. Estas execuções geram uma saída tabular com cabeçalho (-outfmt 7). O valor "*-evalue 1e-10*" informa que uma sequência só será incluída nos resultados se esta tiver um *e-value* menor ou igual a 1e-10. O valor "*-comp_based_stats 0*" informa que não será feita estatística composicional. O valor "*-num_alignments 100000*" informa que puderam ser feitos ate 100000 alinhamentos.

No caso do PSI-BLAST temos ainda os valores "*-inclusion_ethresh 1e-10*" e "*-num_iterations 5*". O primeiro informa que uma sequência só será usada para calcular a próxima matriz de pesos do PSI-BLAST se ela tiver um *e-value* menor ou igual a 1e-10. Já o segundo valor informa que o PSI-BLAST fara no máximo cinco iterações.

Além do BLASTp e do PSI-BLAST também foram usados neste trabalho o *makeblastdb* para se formatar um banco de dados, o *blastdbcmd* para recuperar uma sequência em um banco previamente formatado e o *blastdb_aliastool* para inserir novas sequências em um banco de dados formatado. O *makeblastdb* foi usado da seguinte forma:

```
makeblastdb -in input -parse_seqids -hash_index -out input
```

Onde "*input*" representa o arquivo multi-fasta. O valor "*hash_index*" faz com que o programa crie um índice *hash* que melhora o desempenho do BLASTp e PSI-BLAST. Por fim, pode-se ver que o arquivo de saída tem o mesmo nome do arquivo de entrada.

O "*blastdbcmd*" foi usado das seguintes formas:

```
blastdbcmd -entry identifier -db multi_fasta -out output
```

```
blastdbcmd -entry_batch input_file -db multi_fasta -out output
```

Onde "*identifier*" representa um identificador proteico válido e presente no banco "*multi_fasta*". Já "*input_file*" (linha 2) representa um arquivo texto contendo uma lista de identificadores proteicos válidos e presentes em "*multi_fasta*". Ao se executar a linha 1 o resultado será um arquivo contendo um único fasta e ao se executar a linha 2 o resultado será um arquivo contendo tantos fastas quanto o *blastdbcmd* pode encontra no "*multi_fasta*", dentre aqueles que estão na lista "*input_file*".

O programa *blastdb_aliastool* foi usado da seguinte forma:

```
blastdb_aliastool -dblist "db1 db2" -dbtype prot -out output \
-title title_db
```

Onde "db1" e "db2" são dois bancos multi-fasta formatados, o valor "prot" indica que é um banco de proteínas, "*output*" é o arquivo que conterá o novo banco multi-fasta com a união de "db1" e "db2" e "*title_db*" será o título do novo banco.

Todos os procedimentos de BLAST e gerenciamento de banco de dados fasta foram feitos conforme as linhas de comando acima, exceto quanto explicitado de outra forma.

Os Conceitos de query e subject

Este trabalho não criou conceitos novos para *query* e *subject*, no entanto, cabe uma breve explicação sobre exatamente o que eles significam no contexto deste trabalho. A *query* é uma sequência fasta única usada para a busca de BLASTp ou PSI-BLAST. Neste trabalho não foi feito BLAST multi-query. O *subject* é qualquer sequência encontrada pelo BLASTp ou PSI-BLAST. Em geral a própria *query* é encontrada da busca, logo, nestes casos, a *query* é também um *subject*.

3.3.2 Auto-pontuação

A auto-pontuação é um conceito que foi criado neste trabalho e diz respeito à pontuação que uma proteína atribui a si mesma no PSI-BLAST quando esta é usada em um banco de dados contendo apenas ela mesma. As seguintes linhas de comando BLAST podem ser usadas para se obter a auto-pontuação:

```
makeblastdb -in query -out query
```

```
psiblast -query query -db query -num_iterations 2 \
-evalue 100 -inclusion_ethresh 100 -comp_based_stats 0 \
-out output -outfmt 6
```

Onde "*query*" é um arquivo fasta contendo apenas uma sequência. A primeira linha de comando irá formatar este arquivo gerando os binários que o BLAST reconhece. A segunda linha de comando irá submeter o arquivo "*query*" contra o banco de dados "*query*". Esta execução prosseguirá por duas iterações (num_iterations 2) e retornará um arquivo "*output*" tabular sem cabeçalho (-outfmt 6) contendo quatro linhas.

A primeira linha conterá 12 colunas com os resultados de BLAST relativos ao alinhamento da *query* contra ela mesma na primeira iteração do PSI-BLAST. A segunda linha conterá as mesmas informações da primeira, mas relativas à segunda iteração do PSI-BLAST. As duas linhas seguintes são, uma linha em branco e uma linha informando que a pesquisa BLAST convergiu. A auto-pontuação será a pontuação da *query* na segunda iteração. Ou seja, a 12^a coluna da 2^a linha.

Pontuações relativas

Outro conceito desenvolvido neste trabalho foi a pontuação relativa. Na verdade duas formas diferentes de pontuação relativa (ou relativizada) foram criadas. Na primeira, a pontuação é relativizada pela auto-pontuação da *query*. Neste procedimento, a auto-pontuação é

calculada para a *query* e em seguida todas as pontuações da tabela de resultados do BLASTp ou PSI-BLAST são divididas por este número (a auto-pontuação da *query*).

Na segunda forma de relativização, foi calculada auto-pontuação para cada *subject* que o BLASTp ou PSI-BLAST recrutou. Então as pontuações destes *subjects* foram divididas por suas respectivas auto-pontuações.

Auto-pontuação relativa

Por fim, um último conceito de pontuação desenvolvido neste trabalho foi a auto-pontuação relativa. Esta pontuação é na verdade um subconjunto de pontuações relativas e será mais fácil explicá-la como subconjunto de pontuação normalizada pela *query*. Conforme já foi dito, uma forma de relativizar as pontuações dos *subjects* (incluindo a própria *query*), é dividir todas estas pontuações por um mesmo número, a auto-pontuação da *query*. Quando o *subject* em questão é a própria *query*, o resultado desta divisão é a auto-pontuação relativa.

A criação do conceito de auto-pontuação relativa baseia-se no fato de que a pontuação de PSI-BLAST é calculada utilizando-se uma matriz de substituição PSSM. No alinhamento da *query* contra ela mesma, todas as posições estarão alinhadas e a matriz ponderará a chance de dois resíduos serem observados alinhados pela chance ao acaso, esta última calculada a partir da frequência de uso do aminoácido. Durante as iterações do PSI-BLAST, é possível que proteínas não homólogas às *queries* passem a ser incorporadas em grande quantidade, a ponto da *query* começar a receber pontuações insignificantes pela PSSM gerada. Assim, a queda do parâmetro auto-pontuação relativa da *query* revela possível deterioração da matriz.

3.3.3 Algoritmos de Aprendizado de Máquina

Engenharia de Features

Neste trabalho o termo *feature* foi traduzido como atributo. A partir deste ponto do trabalho será usada esta palavra.

O procedimento de engenharia de atributos (ou engenharia de *features*) descrito aqui é na verdade o processo de buscar a informação que esta dispersa na tabela de resultados do BLASTp ou PSI-BLAST. Três grupos de informações foram selecionados. O primeiro foi baseado na pontuação do BLAST (*bit-score*), o segundo foi baseado na pontuação da própria *query* e o terceiro foi o tamanho da *query*.

Os PSI-BLASTs feitos neste trabalho têm cinco iterações, logo, a princípio, cada *subject* possui cinco pontuações, uma em cada iteração. Estas pontuações foram normalizadas pela auto-pontuação da *query* e também pela auto-pontuação do *subject* gerando assim 10 pontuações. Uma 11^a "pontuação" foi calculada medindo-se a média das outras 10. Estas 11 pontuações serão chamadas deste ponto em diante de "atributos baseados em pontuação relativa".

Antes de se passar para o próximo grupo de atributos criados, algumas considerações precisam ser feitas. Em primeiro lugar, o *subject* não necessariamente é recrutado em todas as cinco iterações do PSI-BLAST. Neste caso, nas iterações em que ele não aparece, é atribuído a ele uma pontuação zero. Em outras palavras o *subject* é tratado como se ele estivesse lá e tivesse pontuação zero. Em segundo lugar, o PSI-BLAST pode convergir com menos de cinco iterações.

Neste caso todos os *subjects* recebem pontuações zero nas iterações que não foram feitas. Em terceiro lugar, o *subject* pode fazer mais de um alinhamento com a *query*, sendo assim recrutado mais de uma vez na mesma iteração. Neste caso a primeira aparição, que tem pontuação mais alta, foi usada.

O segundo grupo de informação consiste em uma adição aos vetores acima: a pontuação relativa da *query* ao longo das cinco iterações. Neste caso um único vetor em $\mathbb{R}^5$ foi criado com as cinco pontuações que a *query* atribui a si mesma nas 5 iterações, normalizadas por sua auto-pontuação. Dentro de um PSI-BLAST estes cinco valores serão sempre os mesmos, mas eles variam entre PSI-BLASTs. Estes cinco valores foram concatenados com cada um dos vetores em $\mathbb{R}^{11}$ de atributos baseados em pontuação relativa, criando se assim um vetor em $\mathbb{R}^{16}$. Estes cinco valores serão chamados de atributos baseados em auto-pontuação relativa.

Algumas considerações precisam ser feitas a respeito dos atributos baseados em auto-pontuação relativa. Caso a *query* não seja recrutada em alguma iteração ela recebe pontuação zero e caso o PSI-BLAST convirja antes de cinco iterações, a *query* recebe pontuação zero nas iterações que não foram feitas.

Por fim, o logaritmo do tamanho da *query* (número de aminoácidos) foi calculado e adicionado ao vetores em $\mathbb{R}^{16}$, criados anteriormente, gerando-se assim vetores em $\mathbb{R}^{17}$.

O algoritmo 1 mostra a extração dos 17 atributos a partir do um resultado de um PSI-BLAST. Como se pode ver neste algoritmo na linha 1 é criado uma tabela do tipo chave-valor onde a chave é o identificador da proteína e o valor é o vetor de atributos. A linha 4 inicializa o contador de iterações de PSI-BLAST e as linhas 8-11 incrementam este contador. As linhas 11-24 leem as pontuações de PSI-BLAST e alocam estes valores no vetor de atributos. Como se pode ver os valores são alocados aos pares e sua posição no vetor é dada pela iteração de PSI-BLAST em que o valor foi encontrado. Nas linhas 25-27 as auto-pontuações relativas são salvas em um vetor à parte. Nas linhas 31-44 os cinco pares de pontuações dos vetores de atributos são relativizados pelas auto-pontuações da *query* e dos *subjects*. Nas linhas 38 e 40 a média destas 10 pontuações é calculada e adicionada ao vetor. Na linha 37 as auto-pontuações relativas são adicionadas ao vetor de atributos e na linha 44 o logaritmo na base 10 do número de aminoácidos da *query* é adicionado ao vetor de atributos.

Este algoritmo explica como extrair os atributos de um arquivo PSI-BLAST. Para se extrair atributos de múltiplas saídas PSI-BLAST este algoritmo deve ser aplicado a cada PSI-BLAST e os resultados devem então ser concatenados em uma matriz única.

```
   Data: query, blast_file, mult_fasta_db
   Result: a table with pairs subjects, features
 1 table = empty table of pairs subject, features;
 2 self_scores = an array of zeros in ℝ^5;
 3 mean = 0;
 4 i = 0;
 5 j = 1;
 6 while Not at the end of blast_file do
 7     read current line;
 8     if line starts with "#" then
 9         i = j + 1;
10         j = 0;
11     else
12         extract 'subject' and 'score' from line;
13         if table contains subject then
14             features = table[subject];
15             if features[2*i - 1] == 0 then
16                 features[2*i - 1] = score;
17                 features[2*i] = score;
18             end
19         else
20             features = an array of zeros in ℝ^17;
21             features[2*i - 1] = score;
22             features[2*i] = score;
23         end
24         insert pair 'subject, features' into 'table';
25         if subject == query & self_cores[i] == 0 then
26             self_scores[i] = score/calculate_self_score(subject, mult_fasta_db);
27         end
28         j = 1;
29     end
30 end
31 foreach pair 'subject, features' in table do
32     subject_score = calculate_self_score(subject, mult_fasta_db);
33     query_score = calculate_self_score(query, mult_fasta_db);
34     for i = 1 to 5 do
35         features[2*i - 1] = feature[2*i - 1]/query_score;
36         features[2*i] = feature[2*i]/subject_score;
37         features[11 + i] = self_scores[i];
38         mean = mean + feature[2*i - 1] + feature[2*i];
39     end
40     feature[11] = mean;
41     mean = 0;
42     insert pair 'subject, features' into 'table';
43 end
```

Algorithm 1: Algoritmo mostrando o processo de criação dos 17 atributos. Neste algoritmo "table" é uma tabela tipo chave-valor, onde chave é o identificador do subject e o valor os atributos. Nas linhas 6-30 esta tabela é preenchida com as pontuações de BLAST. Nas linhas 31-43 estas pontuações são normalizadas e o average_score (mean) e os self_scores são adicionados aos atributos (features). Este código pressupõe a existência de uma função capaz de receber o identificado de uma proteína e o banco multi-fasta e calcular sua auto-pontuação.

Rede Neural

A rede neural usada neste trabalho foi a da biblioteca *nnet* de R. Essa biblioteca implementa uma MLP com uma camada oculta e função de ativação sigmoide. Para se utilizar essa rede ela foi carregada no ambiente R e o treinamento foi feito conforme a seguinte linha de comando:

```
model <- nnet(
                x=features,
                y=response,
                size=num_neurons,
                entropy=TRUE,
                maxit=400
        )
```

Onde "x=features" é a sintaxe para passar a matriz de atributos, sem as respostas, para a rede; "y=response" passa para a rede as respostas (0 ou 1); "size=num_neurons" diz quantos neurônios na camada oculta devem existir; "entropy=TRUE" informa que treinamento deverá ser feito com entropia ligada; e "maxit=400" diz que o máximo de iterações (ou épocas) a serem feitas será 400. O objeto com a rede treinada é então salvo com o nome "model". Essa variável pode ser usada mais tarde para fazer predições. Todas as redes, exceto aquelas com zero neurônios na camada oculta, foram treinadas desta forma. Já as com zero neurônios foram treinadas com a seguinte linha de comando.

```
model <- nnet(
                x=features,
                y=response,
                size=0,
                skip=TRUE,
                entropy=TRUE,
                maxit=400
        )
```

Esta linha é quase idêntica à anterior, exceto que "*size*" é sempre igual a zero e "*skip=TRUE*", que instrui a rede a não criar uma camada oculta. Por padrão este parâmetro é falso.

Random Forest

Neste trabalho foi usada a biblioteca "*randomForest*" de R. Esta biblioteca foi carregada no ambiente R e para se treinar os modelos de *random forest* foi usada a seguinte linha de comando:

```
model<-randomForest(
                x=features,
                y=response,
                ntrees=num_tree,
                mtry=num_features
        )
```

Onde "x=features" e "y=response" são a matriz de atributos e o vetor com respostas assim como no *nnet*. Já o valor "ntree=num_trees" informa quantas árvores de classificação devem ser criadas. O valor "mtry=num_features" informa quantos atributos devem ser selecionados para a criação de cada árvore. E "*model*" é o objeto com o modelo que pode ser usado posteriormente para predições.

Predição e Medidas de qualidade

Para se fazer predições neste trabalho foi usada a função *predict* de R, conforme a seguinte linha de comando:

```
predictions<-predict(model, features)
```

Onde "*model*" é um objeto como o modelo treinado. Este objeto pode ser tanto da classe *nnet*, quanto da classe *randomForest*. O valor "*features*" representa uma matriz em $\mathbb{R}^{mx17}$ com os atributos e a saída "*predictions*" representa as predições que são feitas. Estes valores são em geral valores entre 0 e 1.

Para se medir a qualidade da predição foi utilizada a AUC (Area Under the Curve) que mede a área abaixo da curva ROC (Receiver Operating Characteristic). Para se medir a AUC a função *roc.area* da biblioteca *verification* de R foi usada, conforme a seguinte linha de comando.

```
roc.area(real, predictions)$A
```

Onde "*real*" são as respostas binária de casa *subject* e "*predictions*" são as predições feitas pelo modelo. Esta função imprime na tela a AUC junto com outros valores. A marcação "$A" informa a função para imprimir apenas o atributo "A" que neste caso é a AUC.

Alem da AUC em alguns experimentos foram medidas precisão e revocação (*recall*). A primeira métrica mede a porcentagem de verdadeiros positivos retornado por uma modelo ou predição. Já a segunda mede a porcentagem do total de positivos recuperados pelo modelo.

3.3.4 Testes e Medidas Estatísticas

Neste trabalho foi feito teste da media com teste-t de Student. Para se fazer estes testes foi usada a função "t.test" do software estatístico R.

Também foi usado correlação de Pearson em diferentes momentos do trabalho.

3.3.5 Distância e Árvore Filogenética

Para determinar as distâncias filogenéticas da query contra as proteínas recuperadas pelo Annothetic, todas as sequências proteicas são primeiramente submetidas ao alinhador múltiplo MUSCLE. Posteriormente, o alinhamento múltiplo gerado é submetido ao programa FastTree v.2.1.7, que utiliza métodos heurísticos para gerar uma árvore filogenética de máxima veros-similhança com grande número de sequências. A árvore gerada é então processada por um parser que calcula o comprimento dos ramos entre a query e as demais sequências amostradas na árvore. O parser foi escrito na linguagem PERL e utiliza pacotes do BioPerl.

3.3.6 Segment Overlap

Para determinar a sobreposição de estruturas secundárias, partiu-se da extração de segmentos estruturados utilizando o algoritmo DSSP. Os resultados de DSSP para todo o PDB foram descarregados pelo endereço `ftp://ftp.cmbi.ru.nl//pub/molbio/data/dssp/` e a partir dele foram gerados arquivos em formato fasta em que as estruturas alfa hélice, fita beta e desordenada foram codificadas por letras H, E, e C, respectivamente. O parâmetro SOV (Segment Overlap) foi calculado por uma função escrita em MATLAB que realiza primeiramente um alinhamento global entre as sequências das estruturas secundárias com o software PRANK Loytynoja and Goldman (2010) para em seguida calcular as medidas de sobreposição.

3.3.7 Lowest Common Ancestor

O LCA (Lowest Common Ancestor) é uma técnica que encontra o menor ancestral comum entre um grupo de organismos. Esta técnica também é capaz de medir em qual nível da árvore da vida este ancestral esta (LCA-*level*). Neste trabalho para se fazer o cálculo de LCA os taxids do *subject* e da *query* são obtidos na tabela idmapping. Com estes taxids a seguinte consulta SQL é feita:

```
SELECT * FROM tax_simple WHERE genome_tax_id = taxid;
```

Essa consulta retorna uma lista com 18 valores onde cada valor é um id de taxon (p. ex.: id da espécie, id do gênero, etc) e cada posição da lista é um nível de taxid (p. ex.: espécie, gênero, etc). Essa consulta é feita para a *query* e para o *subject*. As duas listas (resultados das consultas SQL) são percorridas até que se encontre o primeiro valor em comum entre as duas. Este valor é o LCA e o índice deste valor é o LCA-*level*.

Está claro no parágrafo acima que neste trabalho o LCA foi calculado entre pares de organismos, a saber: o organismo da *query* e o organismo do *subject*.

3.4 Treinamento e Teste de Modelos Parte 1: PSI-BLAST

3.4.1 Criação dos Conjuntos de Dados de Treino e de Teste

Foram selecionadas quatro listas de *queries* a partir da base de dados PANTHER. A primeira destas listas foi um conjunto de 1200 *queries* selecionadas aleatoriamente. E as outras três são oriundas de *clusteres* PANTHER completos. Sendo que a primeira delas vêm de *clusteres* com menos de 100 proteínas, a segunda vem de *clusteres* com tamanho entre 100 e 1000 proteínas e a terceira é o maior *cluster* com 1583 proteínas.

O grupo de 1200 proteínas PANTHER aleatórias foi dividido em dois subgrupos, sendo um com 800 proteínas e outro com 400. Desta forma foram criados cinco conjuntos de proteínas e cada um destes foi submetido a PSI-BLAST (3.3.1) contra a base de dados PANTHER-UniProt. Os procedimentos descritos na seção 3.3.3 foram submetidos a cada um destes cinco conjuntos de PSI-BLASTs gerando cinco matrizes de atributos.

Amostragem dos Conjuntos de Dados

As cinco matrizes criadas, conforme explicado anteriormente, foram carregadas no ambiente R. De cada uma destas matrizes foram tomadas duas amostras. Para a matriz de 800 *queries* aleatórias estas amostragens foram seleções de 200 mil instâncias feita de maneira aleatória e sem reposição. Para as outras quatro matrizes o processo foi mais sofisticado. Primeiramente as instâncias foram separadas conforme seu PSI-BLAST de origem. Em seguida, matriz foi dividida em duas, de forma que, cada matriz contivesse metade dos PSI-BLASTs. A partir dai, uma amostra sem reposição de 50 mil instâncias foi retirada de cada uma das duas. As duas matrizes de 200 mil instâncias oriundas das 800 *queries*-aleatórias serão chamadas de ”treino-1” e ”treino-2”. E as duas amostras de 50 mil instâncias oriundas das 400 *queries* aleatórias serão chamadas de ”teste-1” e ”teste-2”.

Tabela 3.1: Tabela mostrando o esquema de uso das matrizes para treino e teste. As amostras ’treino-1’ e ’treino-2’ foram usadas para treino. Cada treino foi testado na própria amostra bem como nas amostras ’teste-1’ e ’teste-2’. Foram feitos quatro treinos, dois em cada amostra de treino, e oito testes, quatro em cada amostra de teste (teste-1 e teste-2).

Treino	1^{st}-Teste	2^{sd}-Teste
Treino-1	Treino-1	Teste-1
	Treino-1	Teste-2
Treino-1	Treino-1	Teste-1
	Treino-1	Teste-2
Treino-2	Treino-2	Teste-1
	Treino-2	Teste-2
Treino-2	Treino-2	Teste-1
	Treino-2	Teste-2

De posse destas 10 amostras os treinos e testes foram feitos. A tabela 3.1 mostra como cada matriz foi usada para treino e para teste. Essa tabela se aplica a todos os treinamentos de redes-neurais e *random forest* feitos neste trabalho. Como se pode ver nesta tabela foram feitos duas replicadas de treino, uma usando ”treino-1” e outra usando ”treino-2”, e cada treino foi

testado em todas as amostras, inclusive no próprio treino.

3.4.2 Redes Neurais

Como se pode ver nas duas linhas de comando na seção 3.3.3, o único parâmetro que está sendo trabalhado é o número de neurônios na camada oculta (*size*). Nestes ensaios foram testadas redes com 0, 2, 4, 8, 12, 16, 20, 25, 30 e 40 neurônios. Cada uma destas condições foi treinada e testada de acordo com o esquema da tabela 3.1.

Neste trabalho também foi feito uma forma simples de *ensemble* de redes neurais. Neste *ensemble*, várias redes com arquitetura idêntica foram treinadas com o mesmo conjunto de dados. Nos testes, cada rede fez uma predição e a predição do ensemble foi a média das predições.

Para os experimentos de *ensemble* foram treinadas 10 redes com cada uma das seguintes arquiteturas: 0, 2, 4, 8, 12 neurônios. Então, foram tiradas as médias das predições de duas redes, três redes, quatro redes e assim por diante até chegar às 10 redes, gerando assim nove condições de *ensemble* para cada arquitetura. Todo este processo foi então replicado segundo o esquema da tabela 3.1.

Para cada uma dos experimentos descritos nesta seção foi medida a AUC nos dados de teste (ver tabela 3.1). As AUCs oriundas de replicatas do mesmo experimento foram então representados na forma de média e desvio padrão.

3.4.3 Random Forest

Como se pode ver na linha de comando da seção 3.3.3, a *random forest* que está sendo testada possui dois parâmetros, sendo eles o número de árvores (*ntree*) e o número de atributos por árvore (*mtry*).

Foram treinadas *random forests* com 1, 10, 25, 50, 75, 100, 125, 150, 175 e 200 árvores com número de atributos por árvore igual a seis. Este valor de *mtry* igual a seis é o padrão para *random forest* com 17 atributos. Para cada uma destas 10 condições testadas o esquema de replicata da tabela 3.1 foi usado.

Também foi avaliado o parâmetro *mtry*. Foram treinadas *random-forests* com *mtry* igual a 1, 2, 3 e 6 e número de árvores fixo em 100. Novamente cada uma destas quatro condições foi replicada pelo esquema da tabela 3.1.

Para cada um dos experimentos descritos nesta seção foi medida a AUC nos dados de teste (ver tabela 3.1). As AUCs oriundos de replicatas do mesmo experimento foram então representados na forma de média e desvio padrão.

3.4.4 Ensemble de Random Forest e Redes Neurais

Foi feito o treinamento e *ensemble* de quatro redes neurais com oito neurônios na camada oculta. Este *ensemble* foi feito conforme já descrito na seção 3.4.2 e será chamado a partir deste ponto de modelo de redes-neurais. Também foi treinada *random forest* com 75 árvores e dois atributos por árvore; este modelo será chamado de modelo de *random forest*. Testamos então o *ensemble* destes dois modelos, sendo que este *ensemble* foi feito de duas maneiras diferentes

através da média simples e de regressão linear. O *ensemble* feito com média simples foi replicado conforme o esquema da tabela 3.1, mas o ensemble baseado em regressão linear não.

O *ensemble* com média é obtido calculando-se a média das predições dos dois modelos. Já no *ensemble* com regressão linear a predição é obtida submetendo-se as duas predições a um modelo linear que pondera a importância de cada um conforme a equação abaixo:

$$. \boldsymbol{y} = a1 * \boldsymbol{rf} + a2 * \boldsymbol{net} + a3 \tag{3.1}$$

Onde "rf" e "net" são, respectivamente, os vetores com as predições da rede neural e da *random forest*. Enquanto "a1", "a2" e "a3" são parâmetros do modelo.

Para se treinar o modelo linear as predições feitas em uma das amostras de 50 mil instâncias do teste foram usadas para treinar o modelo linear e a outra amostra de 50 mil foi usada para teste. A replicata neste modelo foi feita da seguinte forma: primeiro o ensemble foi treinado em uma amostra de 50 mil instâncias de teste e testado na outra amostra de 50 mil e em seguida ele foi treinado na segunda amostra e testado na primeira.

Teste com Clusteres PANTHER Completos

Além dos testes descritos acima o modelo de *ensemble* por média também foi testado nas amostras das matrizes criadas com as *queries* de *clusteres* PANTHER completos. O esquema de teste pode ser visto na tabela 3.2.

Assim como nas seções anteriores, foi medido os AUCs dos experimentos aqui descritos e aqueles vindos de replicatas do mesmo experimento foram mostrados na forma de média e desvio padrão.

Tabela 3.2: Tabela mostrando o esquema de teste em *clusteres* PANTHER completos. A primeira coluna exibe as amostras que foram usadas para se treinar o *ensemble*; estas amostras são as mesmas já descritas enteriormente. As colunas 2, 3 e 4 mostram as amostras de *clusteres* PANTHER completos que foram usadas para teste. Foram feitos 4 teste em cada amostra de PANTHER completo.

Treino	1^{st}-Teste	2^{sd}-Teste	3^{th}-Teste
Treino-1	grande-1	médio-1	pequeno-1
	grande-2	médio-2	pequeno-2
Treino-1	grande-1	médio-1	pequeno-1
	grande-2	médio-2	pequeno-2
Treino-2	grande-1	médio-1	pequeno-1
	grande-2	médio-2	pequeno-2
Treino-2	grande-1	médio-1	pequeno-1
	grande-2	médio-2	pequeno-2

3.4.5 Evolução do Modelo

Além dos treinamentos de *rando forest* com todos os 17 atributos, feitos na seção 3.4.3, também foram feitos treinamentos utilizando-se apenas subconjuntos destes atributos. Os seguintes subconjuntos foram testados:

1. os cinco atributos baseados em pontuação relativa; normalizada pela auto-pontuação da *query*;

2. os 10 atributos baseados em pontuação relativa, normalizados tanto pela auto-pontuação da *query* quanto do *subject*;

3. os 15 atributos formados pela união dos 10 atributos do descritos no item anterior com os cinco atributos baseados em auto-pontuação relativa;

4. os 16 atributos formados pela união dos 15 anteriores, com o atributo criado com a média dos 10 atributos, baseados em pontuação relativa;

5. Todos os 17 atributos descritos.

Deve-se notar nesta enumeração que ela é incremental. Cada item contem todos os atributos do item anterior mais algum ou alguns atributos.

As *random forests* foram treinadas com cada um destes subconjuntos de atributos seguindo o esquema da tabela 3.1. Os resultados de AUC medidos para replicatas do mesmo experimento foram representados como média e desvio padrão, assim como foi feito nas seções anteriores.

3.4.6 Importância dos Atributos

Foi calculada a importância dos 17 atributos de duas formas diferentes. Na primeira, cada atributo foi usado isoladamente como predição na função descrita na seção 3.3.3 medindo-se assim a AUC de cada atributo. Para se gerar replicatas deste experimento foram criadas oito amostras, sem reposição, de 50 mil instâncias cada, extraídas dos conjuntos "treino-1" e "treino-2" e as AUCs foram medidos nestas amostras.

Na segunda forma foi medida a importância da variável na *random forest*. A partir das *random forests* treinadas segundo a descrição da seção 3.4.3, foi extraído este valor. Para ser mais exato, a importância das variáveis foi extraída de *random forests* com 75 árvores e dois atributos por árvore. A importância da variável é calculada automaticamente pela função "*randomForest*" e estes valores são salvos como um atributo do objeto "*model*" da classe "*randomForest*".

3.4.7 Correlação do Modelo Com Outras Métricas

Foram selecionadas 30 proteínas contendo PDB e estas foram submetidas a PSI-BLAST (seção 3.3.1) e aos procedimentos de extração de atributos (seção 3.3.3). Esses atributos foram usados para se fazer predições usando o modelo de *ensemble* por média simples descrito na seção 3.4.4. Essas predições, que serão chamadas deste ponto em diante de confiança, foram comparadas e plotadas contra as métricas SOV (3.3.6), distância filogenética (3.3.5), LCA (3.3.7), identidade e *e-value*.

Identidade

Para se comparar a identidade com o valor da predição foi usada a identidade calculada pelo PSI-BLAST. Foi usada a identidade da primeira aparição do *subject* no PSI-BLAST. Estes

dois vetores, confiança e identidade, foram carregados no ambiente R e a correlação entre os dois foi calculada com a função *cor*.

E-value

O *e-value* calculado pelo PSI-BLAST foi comparado com o valor da predição. Foi usado o *e-value* da primeira aparição do *subject* no PSI-BLAST. Estes dois vetores, confiança e *e-value*, foram carregados no ambiente R e a correlação entre os dois foi calculada com a função *cor*.

Distância Filogenética

A distância filogenética entre o *subject* e a *query* foi calculada conforme explicado na seção 3.3.5. Os vetores de confiança e distâncias filogenéticas foram carregados no ambiente R e a correlação entre os dois foi calculada com a função *cor*.

Lowest Common Ancestor

O LCA-*level* entre cada *subject* e as *queries* foi calculado conforme explicado na seção 3.3.7. Esses LCA-*leveis* foram agrupados em classes de valor de confiança e a média de cada classe foi calculada. As classes de valor de confiança foram 0.0-0.1, 0.1-0.2, 0.2-0.3, 0.4-0.5, 0.6-0.7, 0.7-0.8, 0.9-1.0.

Segment Overlap

Os *subjects* que possuem PDB foram selecionados e um mapa de PDB contra PDB foi editado (PDBs das *queries* contra PDBs dos *subjects* que elas recrutaram). O SOV entre estes PDB foi calculado usando os procedimentos descritos na seção 3.3.6. Estes SOVs foram agrupados em classes de valor de confiança. A mediana, os quartis e os valores máximos e mínimos foram medidos em cada classe. As classes de valor de confiança foram 0-0.1, 0.1-0.2, 0.2-0.3, 0.4-0.5, 0.6-0.7, 0.7-0.8, 0.9-1.0.

3.5 Treinamento e Teste de Modelos Parte 2: BLASTp

3.5.1 Criação dos Conjuntos de Dados de Treino e de Teste

As 1200 *queries* selecionadas na seção 3.4.1 e que foram divididas em uma lista de 800 e outra de 400 proteínas, foram usadas para se criar os conjuntos de treino e teste para os modelos de BLASTp. Para isso cada uma destas duas listas foi submetida a BLASTp e os procedimentos de extração de atributos descritos na seção 3.3.3 foram aplicados a estes dois conjuntos de BLASTps gerando duas matrizes de atributos.

Tabela 3.3: Tabela mostrando o esquema de uso das matrizes para treino e teste. As amostras 'treino-1' e 'treino-2' foram usadas para treino. Cada treino foi testado na própria amostra bem como nas amostras 'teste-1' e 'teste-2'. Foram feitos quatro treinos, dois em cada amostra de treino, e 8 testes, quatro em cada amostra de teste (teste-1 e teste-2).

Treino	1^{st}-Teste	2^{sd}-Teste
Treino_p-1	Treino_p-1	Teste_p-1
	Treino_p-1	Teste_p-2
Treino_p-1	Treino_p-1	Teste_p-1
	Treino_p-1	Teste_p-2
Treino_p-2	Treino_p-2	Teste_p-1
	Treino_p-2	Teste_p-2
Treino_p-2	Treino_p-2	Teste_p-1
	Treino_p-2	Teste_p-2

Estas duas matrizes foram submetidas a procedimentos de amostragem idênticos àqueles descritos na seção 3.4.1. Isso gerou as matrizes mostradas na tabela 3.3. Essa tabela mostra como cada matriz foi usada para treino e para teste. A tabela 3.3 se aplica a todos os treinamentos de redes-neurais e *randomForest* feitos neste trabalho com dados de BLASTp.

3.5.2 Rede Neural

Foram testadas redes neurais com zero, dois, três, quatro e cinco neurônios na camada oculta. Estes testes foram feitos seguindo o esquema da tabela 3.3. Para cada uma destes experimentos foi medido o AUC nos dados de teste (ver tabela 3.3). Os dados de AUC oriundos de réplicas do mesmo experimento foram então representados na forma de média e desvio padrão.

3.5.3 Random Forest

Foi treinada *random forest* com 200 árvores e 1 atributo por árvore. Este experimento foi replicado segundo o esquema da tabela 3.3. A AUC de cada réplica foi medida e a média e desvio padrão foram calculados.

3.5.4 Ensemble de Random Forest e Rede Neural

Foi feito o *ensemble* dos resultados da rede neural com quatro neurônios na camada oculta com os resultados da *random forest*. O *ensemble* foi feito de duas maneiras distintas. Primeiro com a média simples dos resultados, segundo com regressão linear (3.1). O *ensemble* feito com

média simples foi replicado conforme o esquema da tabela 3.3, mas o *ensemble* baseado em regressão linear não.

Para se treinar o modelo linear as predições feitas em uma das amostras de 50 mil instâncias do teste foi usada para treinar o modelo linear e a outra amostra de 50 mil foi usada para teste. A réplica neste modelo foi feita da seguinte forma: primeiro o *ensemble* foi treinado em uma amostra de 50 mil instâncias de teste e testado na outra amostra de 50 mil instâncias e em seguida ele foi treinado na segunda amostra e testado na primeira.

Assim como nas seções anteriores foram medidas as AUCs dos experimentos aqui descritos. Aqueles vindos de réplicas do mesmo experimento foram mostrados na forma de média e desvio padrão.

3.6 Mineração de Anotação

3.6.1 Criação da Lista de StopWords

Para propósito dos experimentos descritos a seguir, as palavras ”anotação” e ”descrição” quando no contexto de proteína (ex.: ”anotação da proteína”, ou ”descrição de proteína”) serão usadas como sinônimas.

Foi criada uma lista de *stopWords* para ser aplicada em mineração de anotação de proteínas. Para isso, primeiramente, fez-se uma contagem das palavras que ocorrem nas descrições das proteínas do UniProt, versão de fevereiro de 2012. Esta contagem foi feita de maneira a computar em quantas anotações cada palavra ocorre. Portanto, se um termo aparecer duas vezes na mesma descrição ele só será contado uma vez. De posse dessas contagens as palavras que ocorreram mais de 10 mil vezes em todo o UniProt foram selecionadas. Estas palavras foram então checadas manualmente e foi criada a lista de *bio-StopWords* que pode ser vista no apêndice A.

Esta lista foi enriquecida com as palavras que tiveram ”alta” taxa de co-ocorrência com o termo *uncharacterized*. Para se calcular esta co-ocorrência foi usada a equação 3.2:

$$ic = \log_2 \frac{fw12}{fw1 * fw2} \tag{3.2}$$

Onde ”fw1” é a fração de todas as anotações que contêm a palavra ”1”, ”fw2” é a fração das anotações que contêm a palavra ”2” e ”fw12” é a fração das anotações que contêm as duas palavras ao mesmo tempo. E ”ic” logicamente é o índice de co-ocorrência.

As palavras que tiveram índice de co-ocorrência maior ou igual a zero foram usadas para enriquecer a lista de *bio-stopWords*. Esta lista enriquecida foi chamada de *bio-stopWords*-expandida. Essa lista foi ainda enriquecida com as palavras que tiveram contagem menor ou igual a 10 em todo o UniProt. Esta lista duplamente enriquecida será às vezes chamada informalmente de ”baixa contagem” ou de ”*bio-stopWords* duplamente enriquecida”, significando a lista das *bio-stopWords*-expandidas e enriquecida com as palavras com baixa contagem.

Também foi usado em um experimento deste trabalho a lista de *stopWords* do MySQL. Esta lista pode ser encontrada em `http://dev.mysql.com/doc/refman/5.1/en/fulltext-stopwords.html`.

3.6.2 Pré-Processamento de Anotações

Os seguintes pré-processamentos foram aplicados às anotações:

1. Todos os caracteres foram transformados em caracteres minúsculos;
2. remoção de números;
3. remoção de caracteres especiais (caracteres fora da faixa a-z e 0-9);
4. remoção dos números romanos i-xxx.

Em todos os casos acima (exceto o primeiro) foram usadas expressões regulares para se encontrar as sequências que se desejava e transformá-las em espaços em branco. A *string* resultante foi então *tokenizada* por um ou mais espaços em branco.

Todas as anotações receberam os pré-processamentos acima. Além destes procedimentos, diferentes listas de *stop words* foram utilizadas para filtrar palavras pouco informativas. A partir deste ponto do texto chamaremos de pré-processamentos a união dos procedimentos descritos na enumeração acima com a remoção de *stop words* de uma das listas. Deste modo, cinco formas de pré-processamentos foram testadas:

1. sem remoção de *stop words*;
2. remoção de *stop words* de MySQL;
3. remoção de bio-*stop words*;
4. remoção de bio-*stop words*-expandidas;
5. remoção de bio-*stop words* duplamente expandidas.

Lembrando que em todos estes casos os procedimentos com as expressões regulares descritos anteriormente foram aplicados.

3.6.3 Vetor de Bag of Words

Para se comparar as anotações de duas proteínas suas descrições foram primeiro transformadas em *bag of words*. Nesta técnica um vetor, em que cada posição corresponde a uma palavra do vocabulário, é criado e preenchido com os valores "0", caso a palavra não tenha ocorrido naquela descrição, ou "1", caso a palavra tenha ocorrido pelo menos uma vez.

3.6.4 Matriz de Bag of Words

Para se construir uma matriz de *bag of words* a partir do BLAST uma série simples de procedimentos deve ser feita. Primeiramente a tabela de resultados do BLAST deve ser submetido aos procedimentos adequados (BLASTp ou PSI-BLAST) de mineração de atributos e predição (3.4 e 3.5). Feito isso, uma tabela com os *subjects* e seus valores de confiança terá sido gerada. Esta etapa é feita porque alguns experimentos, que serão descritos mais a frente, usaram os valores de confiança calculados pelos modelos de aprendizado de máquina. Esta etapa também tem o efeito positivo de retornar uma lista de *subjects* sem repetição.

Feito isso, as anotações dos *subjects* são adicionadas à lista. Portanto, agora, temos uma lista com *subject*, confiança e anotação. Este ultimo campo é que é de interesse para o procedimento desta seção. Cada anotação é transformada em um vetor de *bag of words* usando os procedimentos da seção 3.6.3. A coleção destes vetores é a matriz de *bag of words*. Os vetores nesta matriz estão sincronizados com os valores de confiança dos *subjects*; para alguns procedimentos isso é importante, para outros não.

O procedimento aqui descrito é genérico e qualquer das listas de *stopWords* ou formas de criação de vetores (*digram*, *uniGram*) poderia ser usada nos métodos acima.

3.6.5 Criação de Vetor-Consenso

O vetor consenso é um vetor com o consenso de todas as anotações. Duas formas de obtenção de vetor consenso foram testadas. A primeira, que é mais simples, ajudará esclarecer o conceito de consenso aqui apresentado.

Seja **M={A_1, A_2, A_3, ..., A_n}** uma matriz formada pelos vetores de *bag of words* $\mathbf{A_1}$ a $\mathbf{A_n}$ criada conforme os procedimentos em 3.6.4. O consenso simples dos vetores em **M** é dado por:

$$. A = \sum_{i=1}^{n} A_{\mathrm{i}} \tag{3.3}$$

Onde **A** é um vetor de mesma dimensionalidade dos vetores $\mathbf{A_1}$ a $\mathbf{A_n}$ e que contém a soma dos valores destes vetores. Esta forma de consenso chamaremos "consenso simples" ou "consenso não ponderado".

A segunda forma de consenso será data pela equação:

$$. A = M^{\mathsf{T}} \cdot f(p) \tag{3.4}$$

Onde $\mathbf{M}^{\top}$ é a transposta da matriz anterior e **f(p)** é uma função que opera sobre o vetor "**p**" que é o vetor de confiança dos *subjects*. O resultado da equação 3.4 é um vetor (**A**) com a soma ponderada dos vetores $\mathbf{A_1}$ a $\mathbf{A_n}$. Para esta forma de consenso é importante a sincronia entre vetores de *bag of words* e confiança do *subject* (ver 3.6.4).

Embora não utilizado neste trabalho, é simples incluir no cálculo do vetor de consenso apenas valores de confiança acima de um certo limiar, como por exemplo 0.5.

Mais adiante neste texto serão mostradas algumas funções '**f(p)**'.

3.6.6 Cálculo de Similaridade de Anotação

Duas formas de "similaridade" de anotação foram usadas neste trabalho. A primeira é a similaridade do cosseno e é uma similaridade de fato. Nesta métrica, o cosseno do ângulo entre dois vetores é medito com a equação 3.5:

$$. \cos\theta = \frac{A^{\mathsf{T}} \cdot B}{||A||||B||} \tag{3.5}$$

Onde "A" e "B" são dois vetores quaisquer em $\mathbb{R}^n$. No denominador estão as normas quadradas dos dois vetores.

A segunda não é propriamente uma similaridade, trata-se da probabilidade de uma palavra presente em uma anotação estar presente em outra anotação. Esta métrica foi aplicada exclusivamente para se comparar a probabilidade de um termo presente na anotação da *query* estar presente na anotação do *subject*.

3.6.7 Criação da Lista de Proteínas Recentemente Anotadas

Foram selecionadas as proteínas do banco de dados UniProt versão de fevereiro de 2012 cuja descrição figurava a palavra *uncharacterized*. Esta lista de proteínas foi filtrada de maneira a se

excluir aquelas cuja anotação tivesse pelo menos uma palavra ausente da lista de *bio-stopWords* duplamente enriquecida, descrita na seção 3.6.1. Esta lista será chamada de 'hipotéticas-2012'.

No UniProt versão de janeiro de 2013 foi feito o processo oposto, foram selecionadas as proteínas cuja anotação tivesse pelo menos uma palavra ausente na lista de *bio-stopWords* duplamente enriquecida. Esta lista será chamada de não hipotéticas-2013.

Tanto a lista hipotéticas-2012 quanto a lista não hipotéticas-2013 são listas de identificadores de proteínas. Foi checado os identificadores que estavam presentes nas duas listas e com estes foi criada a lista de proteínas recentemente anotadas. A partir desta última lista foram selecionadas aleatoriamente 900 proteínas. Esta lista de 900 proteínas recentemente anotadas será chamada de recém-anotadas.

3.6.8 Aplicação do Modelo de Aprendizado de Máquina Sobre a Lista de Recem-Anotadas

Cada proteína da lista de "recém-anotadas" foi submetida a BLASTp e PSI-BLAST, contra UniProt de fevereiro de 2012, e, em seguida, aos procedimentos das seções 3.3.3, e 3.4 ou 3.5. Com estes procedimentos os valores de confiança de cada *subject* recrutados nos BLASTp ou PSI-BLAST foi calculada.

Adicionalmente, as descrições de cada *subject* foram comparadas com as descrições das *queries* que as recrutaram. Nestes testes foram usadas as anotações de janeiro de 2013 para a *query* e de fevereiro de 2012 para os *subjects*. Estas comparações foram feitas de múltiplas formas, mas todas seguiram o seguinte "cronograma": primeiro elas foram pré-processadas (3.6.2), depois elas foram transformadas em vetores de *bag of words* (3.6.3) e, por fim, esses vetores (da *query* e do *subject*) foram comparados através dos métodos descritos em 3.6.6.

Existem duas formas de comparação de anotações, duas formas de criar vetores de *bag of words* e múltiplas formas de pré-processamento. A tabela 3.4 mostra as combinações de técnicas que foram testadas nesta etapa do trabalho.

Cada combinação de técnicas mostrada na tabela 3.4 gerou uma lista de "similaridades" (cosseno ou "probabilidade da palavra"). Estas "similaridades" foram carregadas no ambiente R juntamente com os valores de confiança dos *subjects* e a correlação entre estas duas métricas foi medida com a função *cor*.

Vale lembrar que dados de BLASTp e PSI-BLAST vivem em universos diferentes, portanto, os procedimentos citados acima foram aplicados nestes dois conjuntos de dados separadamente.

3.6.9 Cálculo da Função de Ponderação

Como foi dito na seção 3.6.5 o consenso ponderado tem uma função de ponderação "f(p) ". Nesta seção serão mostradas duas funções de ponderação que foram usadas neste trabalho. Para se encontrar estas funções foram usados os cossenos, entre as anotações das *query* e do *subject*, obtidos com as técnicas "6" da tabela 3.4.

A primeira abordagem foi com dados de PSI-BLAST. Após se aplicar a técnica "6" (3.4) em dados originados de PSI-BLAST, obteve-se uma lista de cossenos, além da lista de valor de confiança, que já havia sido calculada (ver inicio da seção 3.6.8). Os dados de cosseno foram

Tabela 3.4: Tabela mostrando as combinações de técnicas de pré-processamento, criação de vetores de *bag of words*, modo de comparação de similaridade e origem dos dados que foram testados neste trabalho. A coluna 1 da tabela contem um id que servirá para referência ao longo do texto.

Id da Técnica	Técnica de pré-processamento	Técnica de bag-of-words	Técnica de comparação de vetores	Dados a que Foram aplicados
1	sem-stopWords	singleGram	cosseno	PSI-BLAST
2	sem-stopWords	diGram	cosseno	PSI-BLAST
3	MySQL-stopWords	diGram	cosseno	PSI-BLAST
4	bio-stopWords	diGram	cosseno	PSI-BLAST
5	bio-stopWords-expandida	diGram	cosseno	PSI-BLAST
6	bio-stopWords-duplamente enriquecida	diGram	cosseno	PSI-BLAST
7	bio-stopWords-duplamente enriquecida	diGram	confiança	PSI-BLAST
8	bio-stopWords-duplamente enriquecida	diGram	cosseno	BLASTp

então agrupados em classes de 0.5% de valor de confiança, gerando-se assim 200 classes. Para cada classe foi calculada a média dos cossenos. Com os valores de confiança e cosseno médio foi feita regressão para polinômio do tipo:

$$y = a_1 * p + a_2 * p^2 + a_3 * p^3 \tag{3.6}$$

Onde "y" é o cosseno médio, "p" é a confiança e a_1, a_2 e a_3 são parâmetros da regressão. Portanto, criamos uma função de predição da similaridade média do cosseno, entre as anotações da *query* e do *subject*, dada a confiança do *subject*. Este polinômio é uma função que pode ser usada como função de ponderação "f(p) " na seção 3.6.5. É claro que neste caso "p" passa a ser "**p**", o vetor de confiança, e as potências deve ser tratadas como *element-wised*. Em outras palavras cada elemento do vetor e elevado à potência desejada gerando um vetor com mesma dimensionalidade.

Os procedimentos acima foram repetidos com dados de PSI-BLAST processados pelas técnicas "7" (tabela 3.4). Também foi encontrado um polinômio do tipo 3.6, mas neste polinômio "y" é a "taxa de ocorrência da palavra". Foi calculada a correlação entre as predições feitas por estes dois polinômios, mas apenas o primeiro foi usado como função de ponderação.

A segunda abordagem foi feita com as técnicas "8" da tabela 3.4. Neste caso, os dados foram agrupados em classes de 4% de valor de confiança, criando-se 25 classes. Para cada classe foi calculada a média dos cossenos. Com os valores de confiança e cosseno médio foi feita regressão para uma função linear da família:

$$y = a_1 + a_2 * p \tag{3.7}$$

Onde ”y” é o cosseno médio, ”p” é a confiança e a_1, a_2 são parâmetros da regressão. Esta função foi ainda sujeita a posteriore à seguinte restrição: se y for menor que zero o valor zero é imputado em y.

Estas funções foram usadas como funções de ponderação em 3.4, sendo a primeira em dados de PSI-BLAST e a última em dados de BLASTp.

3.6.10 Calculando Cosseno Entre Vetor-Consenso e Vetor-Anotação

Uma vez que se tenha criado um vetor consenso usando alguma das fórmulas da seção 3.6.5 o cosseno entre este vetor e cada vetor anotação (*bag of words*) pode ser calculado (3.5). Existem duas formas de se calcular este consenso (simples e ponderado). Portanto, existem três formas de melhor resultado. A primeira é o *subject* com maior confiança (ver seções 3.4 e 3.5); essa forma não utiliza consenso e será chamada de ”melhor confiança”. A segunda é o *subject* com maior similaridade de cosseno entre seu vetor anotação e o vetor consenso simples; essa forma de melhor resultado será chamada de ”melhor cosseno simples”. E a terceira é o *subject* com maior similaridade de cosseno entre seu vetor anotação e o vetor consenso ponderado; essa forma de melhor resultado será chamada de ”melhor cosseno ponderado”.

3.6.11 Testando o ”Melhor Resutado”

Conforme foi dito anteriormente, três formas de melhor resultado são possíveis: ”melhor confiança”, ”melhor cosseno simples”, ”melhor cosseno ponderado”. Para se comparar estas três formas foi extraído o melhor resultado, por cada um dos três métodos, de cada uma dos 900 BLASTps e 900 PSI-BLASTs feitos nesta seção. Em seguida, o cosseno entre as anotações dos melhores *subjects* e de suas respectivas *queries* foi medido e estatísticas como média e distribuição destes cossenos foram calculadas. Vale lembrar que a anotação dos *subjects* é uma anotação UniProt de fevereiro de 2012 e da *query* é de UniProt de janeiro de 2013. É importante ressaltar também que os experimentos com ’BLASTp’ não foram misturados com experimentos ”PSI-BLAST”.

3.7 Annothetic

Os procedimentos descritos nas seções anteriores foram implementados em um serviço web chamado Annothetic. Este software foi desenvolvido em Java e sua interface web em HTML e PHP. As seguintes funcionalidades estão implementadas neste software: doença

1. BLASTp, PSI-BLAST: o Annothetic dispara estes programas conforme descrito em 3.3.1;
2. cálculo de confiança, tanto para dados de BLASTp quanto PSI-BLAST (seções 3.4 e 3.5);
3. cálculo de LCA conforme descrido em 3.3.7;
4. cálculo de distância filogenética e geração de árvore filogenética conforme descrito em 3.3.5;
5. mineração de anotação pelo método do consenso ponderado conforme descrito em 3.6.5 e 3.6.10.

É importante ressaltar que foi feito um estudo de caso com a proteína "Q9HMT0". Esta proteína pode ser encontrada como "proteína exemplo" no próprio Annothetic (`http://pinguim.fmrp.usp.br/miguel/annothetic/website/index.php?example=1`), bem como no UniProt (`http://www.uniprot.org/uniprot/Q9HMT0`).

3.8 Estudo de Caso com AminoAcil-tRNA Ligases

As 13 aminoacil-tRNA ligases de *Halobacterium salinarium* foram selecionadas e foi feito Annothetic com elas. Os melhores resultados pela confiança e pelo consenso ponderado foram obtidos e comparados.

Também foi feito um cálculo de precisão da busca Annothetic. Neste cálculo foram consideradas todos os resultados independente de confiança ou de qualquer outro valor. Esta precisão foi calculada em dois conjuntos de dados diferentes. Primeiro com *EC-number*, obtidos conforme explicado em 3.2.1. Neste caso, os *EC-numbers* exatamente iguais aos das *queries* foram considerados positivos, *subjects* com *EC-numbers* diferentes daqueles das *queries* foram considerados negativos, e os *subjects* sem *EC-number* foram descartados.

O segundo conjunto de dados foi KO. Os mesmos procedimentos descritos para *EC-number* foram aplicados a KO. Os KOs foram obtidos na tabela "idmapping" (3.2.1).

3.9 Experimento com Quatro Microrganismos

3.9.1 Obtenção das Proteínas Hipotéticas

Conforme descrito na seção 3.2.5, os genomas de quatro micro organismos foram obtidos. Foram filtradas aquelas proteínas com a palavra *hypothetical* na descrição (neste caso descrição KEGG versão de 2010 e não UniProt). Posteriormente os KEGG-ids destas proteínas foram mapeados para UniProt-id (versão de janeiro de 2013) via tabela idmapping. Com estes Ids UniProt as descrições UniProt foram obtidas e filtradas pela lista de *bio-stopWords* duplamente enriquecida (3.6.1) de maneira a se checar quais proteínas continuavam sendo hipotéticas.

3.9.2 Aplicação dos Modelos

Todas estas proteínas foram submetidas as procedimentos descritos nas seções 3.4, 3.5 e 3.6. Ou seja, todas elas foram submetidas a BLASTp e aos procedimentos de cálculo de confiança e mineração de anotação apropriados e todas elas foram submetidas a PSI-BLAST e aos procedimentos de cálculo de confiança e mineração de anotação apropriados. Mais especificamente foi calculado o cosseno entre os vetores anotação do *subject* e o vetor consenso ponderado (conforme descrito em 3.6.10). Nesses experimentos com microrganismos foi usada a ”melhor confiança” como forma de melhor resultado (ver 3.6.10), no entanto, o cosseno também foi calculado, como descrito.

Contagens das Anotações

Para cada microrganismo foi feita uma contagem de quantas proteínas receberam alguma sugestão de anotação com confiança acima de 50% em dados de BLASTp. Também foi contado quantas proteínas receberam alguma sugestão de anotação com confiança maior que 50% em dados de PSI-BLAST.

Por fim, algumas sugestões de anotação feitas pelo modelo de PSI-BLAST foram checadas manualmente e comparadas com possíveis sugestões de anotações feitas pelo eggNOG, GO, Pfam, ProSite e UniProt todos versão de outubro de 2013. Estas comparações foram feitas no endereço `http://uniprot.org`. Este é o endereço do UniProt mas a partir deste site os outros bancos de dados podem ser acessados.

Medindo a Variabilidade das Sugestões de Anotação

Para se medir a variabilidade das anotações encontradas pelo Annothetic para as *queries* dos quatro microrganismos os seguintes procedimentos foram aplicados de maneira incremental:

1. Foram removidas as ”palavras” que misturavam letra e número (p. Ex.: UPF0273) e as palavras com menos de cinco letras. Feito isso foram contadas as anotações que ficaram vazias.

2. Foram removidas as palavras ”outer”, ”inner”, ”integral” e as palavras que continham a *sub-string* ”membrane”, ”cytoplasmic”, ”periplasmic”, ”cytosolic”, ”extracellular”, ”nuclear”. Foram contadas quantas anotações ficaram vazias.

3. Foram removidas as palavras ”lipoprotein”, ”metaloprotein” e ”selenoprotein”. Foram contadas quantas anotações ficaram vazias.

Além destes procedimentos incrementais foram contadas também quantas anotações continham palavras terminadas em ”ase”.

Capítulo 4

Resultados

4.1 Modelos de aprendizado de máquina

4.1.1 PSI-BLAST

Conforme foi dito na introdução (1.4.3), os resultados do PSI-BLAST apresentam uma alta taxa de falsos positivos. Com intuito de filtrar estes resultados construímos um modelo de aprendizado de máquina. Para treinar este modelo fizemos 1200 PSI-BLASTs com *queries* de PANTHER versão de 2010 contra o banco PANTHER-UniProt (Materiais e Método seções 3.4, 3.2.4). Com esses PSI-BLASTs nós criamos 17 atributos a partir das pontuações dos *hits* em cada iteração e do tamanho da query (Materiais e Métodos seção 3.3.3). Com esses atributos nós treinamos os modelos de classificação que serão explicados a seguir.

Redes neurais

O primeiro modelo treinado foi uma rede neural do tipo MLP. Esta rede foi treinada usando os 17 atributos criados a partir de resultados de PSI-BLAST e a saída zero/um de cada *hit* (Materiais e Métodos seções 3.4.2, 3.3.3). A figura 4.1 mostra a variação da AUC no teste e no treino pelo número de neurônios na camada oculta. Pode-se observar neste resultado que foram testadas redes com quantidade de neurônios variando de zero (regressão logística) até quarenta. O gráfico mostra que mesmo uma regressão logística (0-neurônios) foi capaz de atingir AUC próxima a 0.9. Isto se deve ao fato de que os atributos individuais deste modelo (figura 4.7) são fortes, fazendo com que mesmo um modelo simples tenha um resultado razoável. Observa-se também na figura 4.1 que à medida que se aumenta o número de neurônios na camada oculta a diferença entre a AUC do treino e a AUC do teste se torna cada vez maior. Pode-se observar que a AUC do treino aumenta continuamente, saturando eventualmente com 25 neurônios, enquanto que a AUC do teste cresce até 4-12 neurônios e então começa a cair. Esses resultados mostram que, a partir de um certo ponto, o aumento da complexidade da rede não leva a melhoria na predição. Ao invés disso, este aumento de complexidade faz a curva produzir um sobreajuste (*overfiting*) em conformidade com o que foi exposto na introdução (seção 1.1.1).

Um outro resultado que se pode ver na figura 4.1 é o desvio padrão entre as AUCs das redes que foram treinadas. O gráfico mostra que quanto menos neurônios na camada oculta maior a conformidade entre as redes (menor a variância). Este resultado nos leva direto ao nosso próximo experimento como veremos a seguir.

A figura 4.2 mostra o efeito do *ensemble* de redes neurais (ver materiais e métodos seção

3.4.2). Foram calculadas as médias das predições de 1 a 10 redes para cada uma das quatro configurações apresentadas (0, 2, 4 e 8 neurônios). Observa-se na figura que com a regressão logística (0-neurônios) não existe nenhum efeito em calcular a média (fazer *ensemble*) de múltiplas predições. No entanto, nas redes com 2, 4 e 8 neurônios, a média das predições é melhor que a predição individual. A figura 4.2 mostra que ocorre um pequeno porém significativo aumento da AUC quando se aumenta o número de predições de 1 até 4 e a partir deste ponto satura.

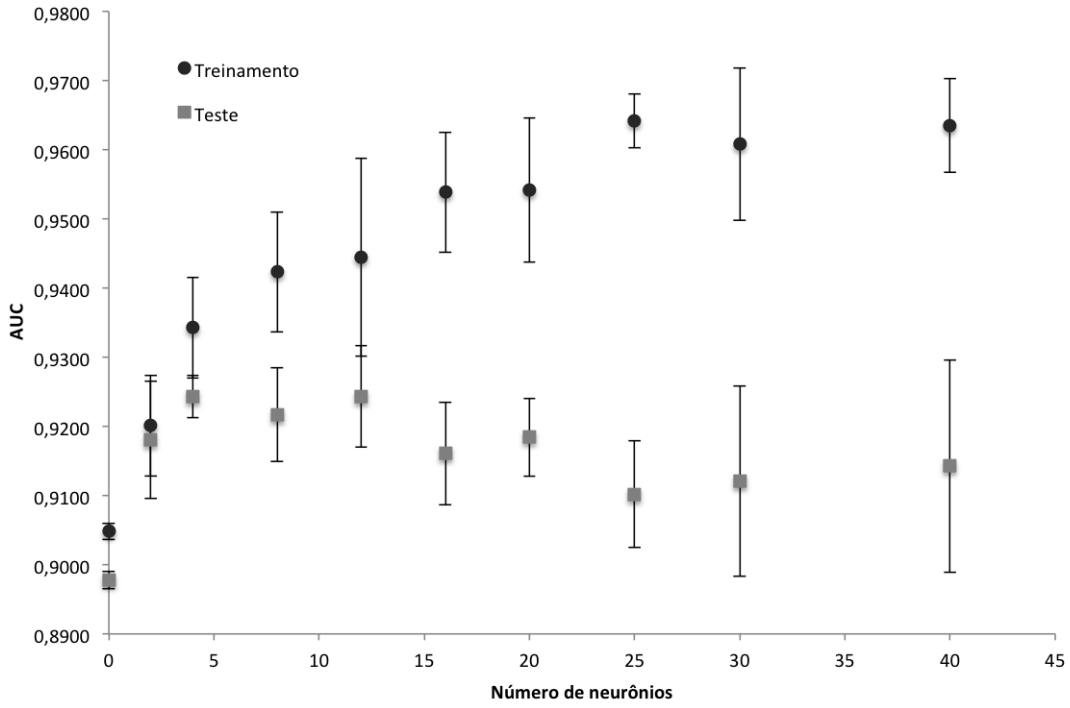

Figura 4.1: Gráfico mostrando a variação do AUC com o número de neurônios na camada oculta. Os pontos quadrados cinza mostram o AUC médio obtido das predições nos dados de teste e os pontos redondos pretos, o AUC médio obtido das predições nos dados de treino. As barras representam o desvio padrão destas médias.

Como foi dito anteriormente, existe um aumento da variância das predições das redes à medida que se aumenta a complexidade das mesmas (figura 4.1). E conforme foi dito na introdução (seções 1.1), a variância é essencial para o aprendizado de máquina. A partir disso depreende-se que o *ensemble* de regressões logísticas não levou a melhoria da predição, uma vez que, os resultados das predições individuais eram muito semelhantes entre si. No entanto, as redes com mais neurônios que geraram predições com maior variação entre si se beneficiaram com o *ensemble* de redes, como é observado na figura 4.2. Além dos resultados mostrados na figura 4.2, também foi testado *ensemble* de redes com mais de 8 neurônios, porém, para estas redes não lograram aumentar a AUC.

A partir da comparação das figuras 4.1 e 4.2, um outro resultado que fica evidente é a redução do desvio padrão quando se faz *ensemble* de múltiplas redes. Esse resultado está em acordo com o que foi dito na seção 1.1.3 da introdução, mostrando que o *ensemble* não só foi

capaz de melhorar a AUC médio, mas também de criar uma predição mais robusta.

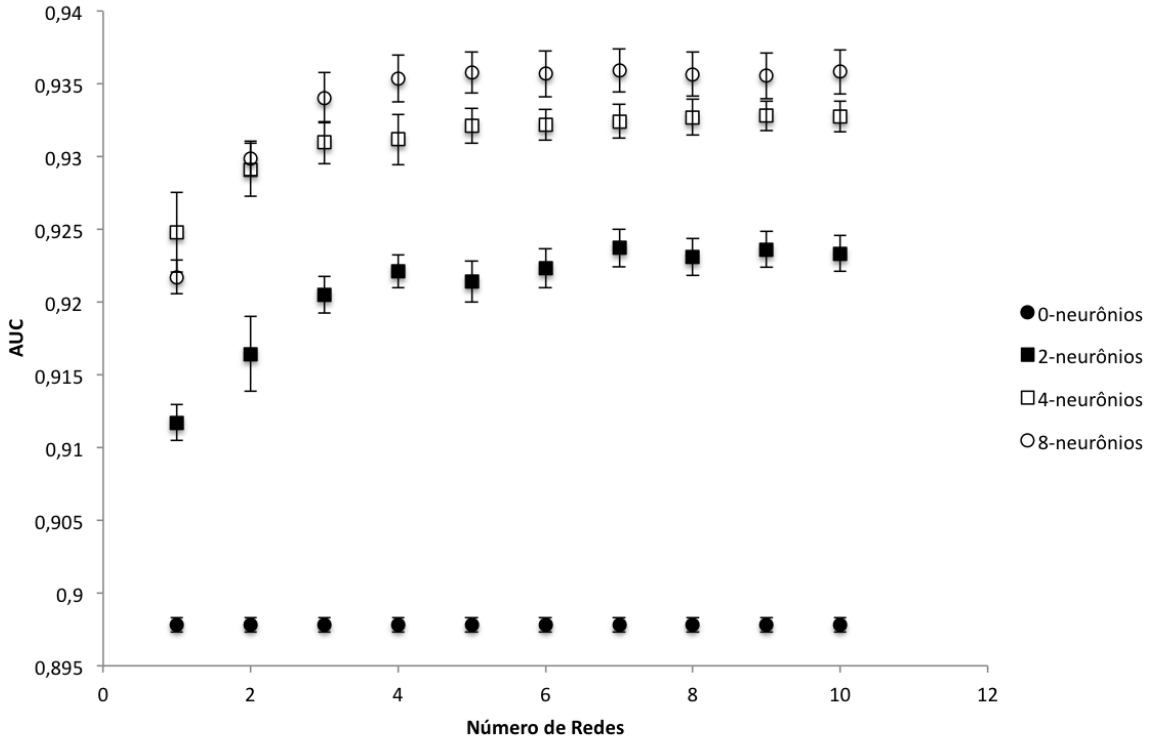

Figura 4.2: Gráfico mostrando a variação da AUC pelo número de redes. O eixo "x" representa o número de redes usadas no *ensemble*. A curva de pontos redondos pretos (mais de baixo) representa *ensemble* de 1-10 redes neurais com zero neurônios na camada oculta; a curva do meio (quadrados pretos) representa o ensemble de 1-10 redes neurais com dois neurônios na camada oculta, a curva de quadrados brancos é o *ensemble* de 1-10 redes com quatro neurônios na camada oculta e os círculos brancos o ensemble de 1-10 redes com oito neurônios na camada oculta. As barras representam o desvio padrão.

A partir dos dados coletados nos experimentos das figuras 4.1 e 4.2, foi determinado que o melhor modelo, para o nosso problema, é o *ensemble* de quatro redes neurais com 8 neurônios na camada oculta. Este modelo alcançou uma AUC de 0.936.

Random Forest (RF)

Nosso segundo modelo foi a *random forest*, que foi feito conforme explicado nas seções 3.3.3 e 3.4.3 dos materiais e métodos. A figura 4.3 mostra a variação da AUC com o número de árvores nos dados de treino e de teste, fixando o número de atributos por árvore em dois. Conforme foi dito na introdução (seções 1.1.3 e 1.1.2) a árvore individual na *random forest* está sujeita a um alto grau de *overfiting* mas o *ensemble* resolve, ou pelo menos reduz este problema. A figura 4.3 corrobora estas observações. Com uma árvore as AUCs no treino e no teste foram respectivamente 0.958 e 0.745, mas com 75 árvores as AUCs foram respectivamente 0.985 e 0.936. Outro dado que se extrai da figura 4.3 são as variâncias. Com uma árvore obteve-se um alto desvio padrão no teste (0.018), mas com o ensemble de 25 ou mais árvores cai para aproximadamente 0.001.

O melhor resultado da *random forest* foi obtido com 75 árvores e a partir deste ponto não houve crescimento significativo. Não obstante, também não houve *overfiting* da forma como ocorreu com a rede neural, ou seja, o aumento do número de árvores não fez a AUC do teste cair.

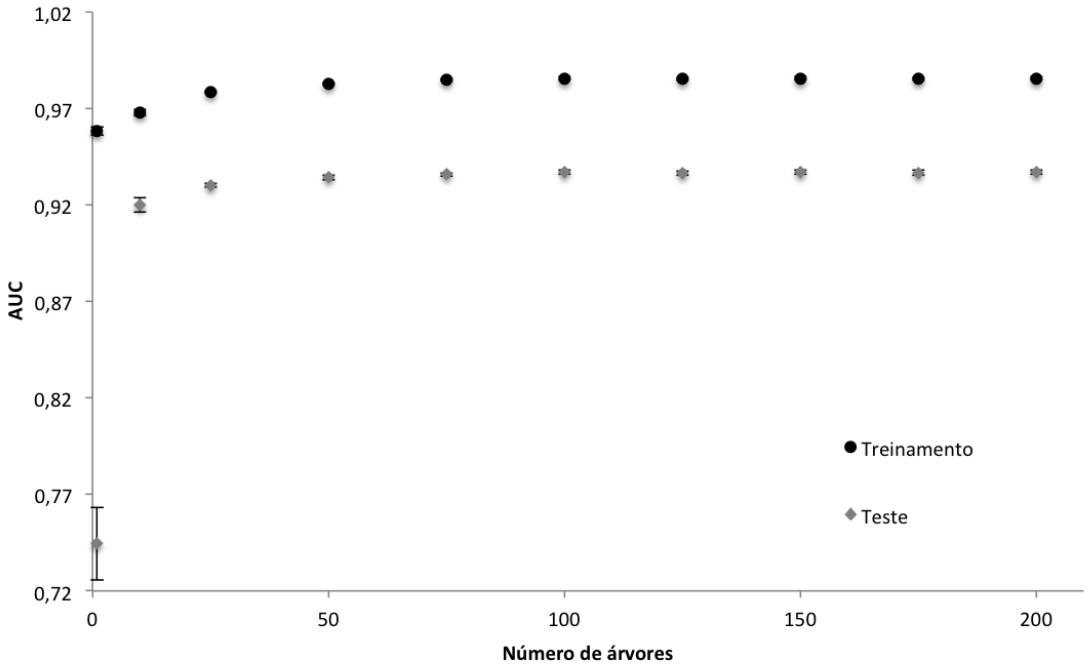

Figura 4.3: Gráfico mostrando a variação da AUC pela quantidade de árvores da *random forest*. Este gráfico foi obtido com *random forests* com dois atributos por árvores. Os pontos em losango cinza mostram o AUC médio obtido das predições nos dados de teste e os pontos redondos pretos o AUC médio obtido das predições nos dados de treino. As barras representam o desvio padrão destas médias.

Um segundo parâmetro da *random forest* é o número de atributos que ela seleciona para construir cada árvore (Materiais e Métodos 3.3.3). De acordo com o que foi exposto na introdução sobre a *random forest* (seção 1.1.2) este parâmetro tem pouca influência sobre a RF. Não obstante, nós testamos o efeito da variação deste parâmetro sobre a RF. A figura 4.4 mostra a

AUC da predição, no conjunto teste, com uma *random forest* com 100 árvores e com atributos por árvore variando de um a seis. O gráfico mostra que o melhor resultado foi alcançado com dois atributos por árvore (0.936) enquanto que com seis, que é o padrão, tivemos o pior resultado 0.932. Embora as diferenças observadas na figura 4.4 sejam pequenas elas são estatisticamente significativa ($p<0.05$).

O fato do melhor resultado ter sido alcançado com apenas dois atributos por árvore se deve ao fato dos atributos individuais serem fortes.

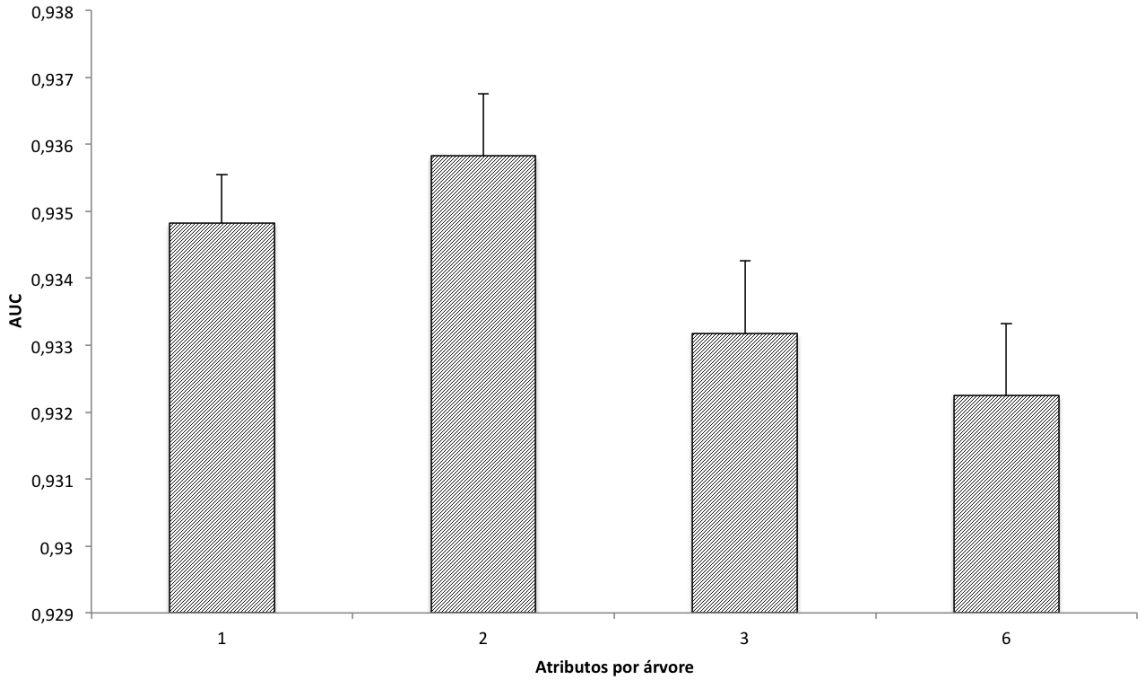

Figura 4.4: Gráfico mostrando a variação do AUC médio pelo número de atributos por árvore em *random forests* com 100 árvores. As barras de erro representam o desvio padrão.

Ensemble de modelos

Uma vez treinados os modelos citados, nós poderíamos compará-los e então selecionar aquele com maior AUC. Ao invés disso fizemos *ensemble* dos dois modelos visando aumentar mais alguns milésimos a AUC. Recapitulando então, fizemos primeiro o *ensemble* de redes neurais, ainda que de forma primitiva, em seguida, *ensemble* de árvores de classificação (com *random forest*). Agora faremos o *ensemble* destes dois modelos para gerar um "super" modelo final.

A figura 4.5 mostra os melhores modelos de *random forest* e redes neurais bem como o *ensemble* destes dois modelos. No gráfico 4.5, "média" representa a média simples das predições dos dois modelos (RNA e RF) e "linear" representa a regressão linear correspondente a uma média ponderada. Observa-se neste gráfico que a média dos dois modelos logrou aumentar a AUC de 0.936 para 0.942, um aumento pequeno porém significativo ($p < 0.05$).

Um fato curioso que se observa no gráfico (figura 4.5), foi que o *ensemble* com média ponderada (regressão linear) não obteve resultado melhor do que aquele com a média simples. No entanto, isso pode ser explicado pelo fato de que os dois modelos tiveram AUCs muito próximos. Mais uma vez, estes resultados corroboram o que foi dito na introdução (seção 1.1.3) sobre *ensemble* de modelos, mostrando que esta técnica é capaz de melhorar as predições em relação aos modelos individuais.

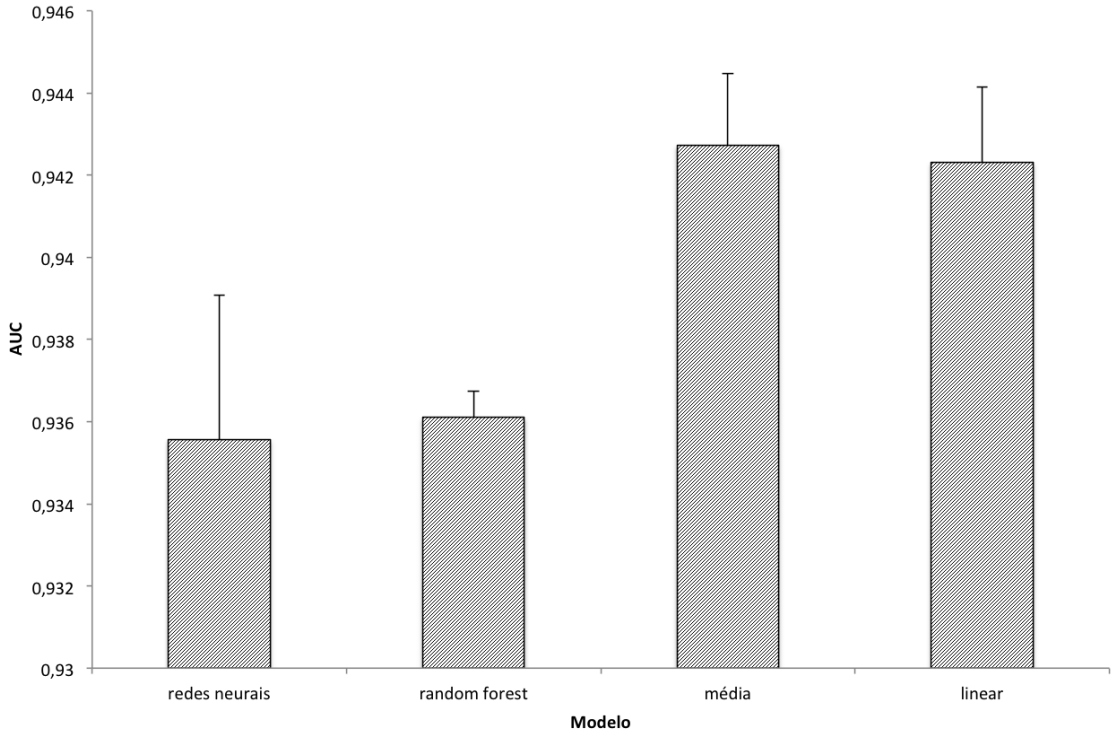

Figura 4.5: Gráfico mostrando a AUC média dos melhores modelos de redes neurais e *random forest* e de dois métodos de *ensemble* destes modelos. As barras de erro representam desvio padrão.

A figura 4.6 mostra a AUC do *ensemble* de modelos e do melhor simples nos dados de teste

(mesmo usado nas figuras 4.1 a 4.4) e em três novos conjuntos: ”panther-grande”, ”panther-médios” e ”panther-pequenos”, e também a média destes três (ver Materiais e Métodos seção 3.4.4). O gráfico mostra uma AUC de 0.985 no ”panther-médios” de 0.963 no ”panther-pequenos” e 0.834 no ”panther-grande”. Esses números mostram que nosso modelo gera predições melhores para proteínas pertencentes a grupos de tamanho pequeno ou médio. Isso reflete o fato de que a maioria das proteínas de PANTHER pertencem a grupos de tamanho médio. Outro fato digno de nota é que a média destas três predições (0.927) foi próxima dos 0.942 do teste. De forma que nestes dados encontramos uma segunda validação do modelo em um novo conjunto de dados, diferente daquele usado no treino e no teste. Embora a diferença entre 0.942 e 0.927 seja estatisticamente significativa, ela pode ser explicada facilmente uma vez que o PANTHER não possui exatamente 33.3% de cada um destes grupos.

Não obstante a melhor predição em grupos médios e pequenos, é importante notar que ainda aí nosso modelo obteve um resultado melhor que o PSI-BLAST sozinho.

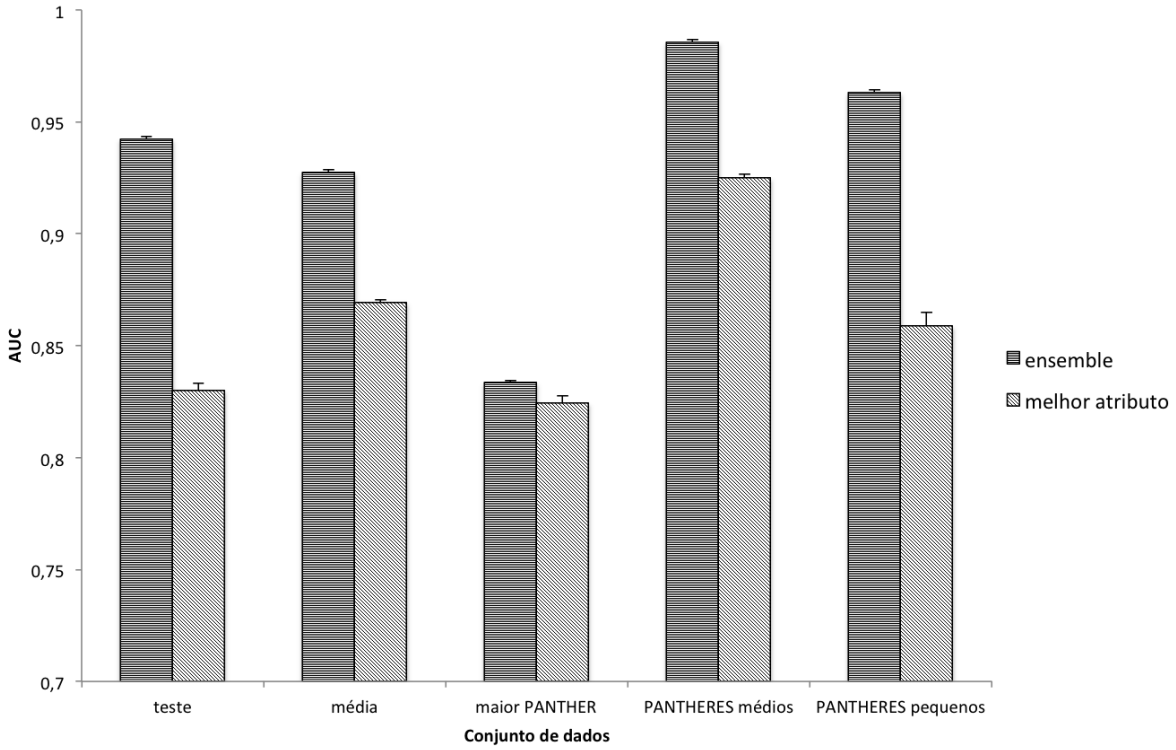

Figura 4.6: Figura mostrando a AUC do modelo final e do melhor atributo simples no conjunto de teste, no maior PANTHER, em PANTHER médios, em PANTHER pequenos e na média destes três. As barras de erro representam desvio padrão.

Atributos

Nessa etapa do trabalho faremos uma análise mais detalhada sobre os atributos, sua importância individual e como eles contribuem para o modelo.

A Tabela 4.1 mostra a correlação dos atributos entre si e com o resultado final. Como se pode observar nesta tabela alguns atributos são fortemente correlacionado. No entanto, nenhum destes é totalmente redundante. E como será visto mais adiante nesta seção todos contribuem para o resultado final do modelo.

Tabela 4.1: Tabela mostrando a correlação dos atributos entre si e com a resposta (binária). Na diagonal da tabela pode-se ver a correlação de cada atributo consigo mesmo. Os valores acima desta diagonal são um "espelho" dos valores abaixo da mesma e estão em cinza.

	QUERY SIZE	QUERY SCORE 1	SUBJECT SCORE 1	QUERY SCORE 2	SUBJECT SCORE 2	QUERY SCORE 3	SUBJECT SCORE 3	QUERY SCORE 4	SUBJECT SCORE 4	QUERY SCORE 5	SUBJECT SCORE 5	AVERAGE SCORE	SELF SCORE 1	SELF SCORE 2	SELF SCORE 3	SELF SCORE 4	SELF SCORE 5	RESPONSE
QUERY-SIZE	1,000	-0,225	-0,151	-0,435	-0,240	-0,453	-0,258	-0,561	-0,315	-0,512	-0,315	-0,426	0,228	-0,624	-0,522	-0,582	-0,542	-0,205
QUERY-SCORE-1	-0,225	1,000	0,910	0,765	0,678	0,651	0,605	0,571	0,528	0,479	0,459	0,791	-0,047	0,167	0,154	0,246	0,293	0,472
SUBJECT-SCORE-1	-0,151	0,910	1,000	0,662	0,752	0,558	0,657	0,469	0,575	0,399	0,496	0,767	-0,028	0,110	0,117	0,199	0,257	0,461
QUERY-SCORE-2	-0,435	0,765	0,662	1,000	0,815	0,779	0,649	0,778	0,639	0,615	0,519	0,882	-0,176	0,273	0,111	0,318	0,405	0,492
SUBJECT-SCORE-2	-0,240	0,678	0,752	0,815	1,000	0,601	0,750	0,564	0,746	0,439	0,584	0,841	-0,135	0,159	0,024	0,206	0,305	0,483
QUERY-SCORE-3	-0,453	0,651	0,558	0,779	0,601	1,000	0,809	0,808	0,635	0,755	0,615	0,873	-0,136	0,282	0,333	0,303	0,391	0,507
SUBJECT-SCORE-3	-0,258	0,605	0,657	0,649	0,750	0,809	1,000	0,612	0,783	0,577	0,743	0,862	-0,115	0,174	0,243	0,220	0,325	0,514
QUERY-SCORE-4	-0,561	0,571	0,469	0,778	0,564	0,808	0,612	1,000	0,766	0,813	0,639	0,853	-0,192	0,360	0,330	0,519	0,539	0,432
SUBJECT-SCORE-4	-0,315	0,528	0,575	0,639	0,746	0,635	0,783	0,766	1,000	0,617	0,804	0,853	-0,177	0,220	0,205	0,389	0,441	0,441
QUERY-SCORE-5	-0,512	0,479	0,399	0,615	0,439	0,755	0,577	0,813	0,617	1,000	0,793	0,781	-0,178	0,319	0,397	0,478	0,629	0,411
SUBJECT-SCORE-5	-0,315	0,459	0,496	0,519	0,584	0,615	0,743	0,639	0,804	0,793	1,000	0,793	-0,179	0,210	0,302	0,381	0,541	0,429
AVERAGE-SCORE	-0,426	0,791	0,767	0,882	0,841	0,873	0,862	0,853	0,853	0,781	0,793	1,000	-0,169	0,278	0,262	0,393	0,497	0,559
SELF-SCORE-1	0,228	-0,047	-0,028	-0,176	-0,135	-0,136	-0,115	-0,192	-0,177	-0,178	-0,179	-0,169	1,000	-0,040	-0,091	-0,188	-0,251	-0,107
SELF-SCORE-2	-0,624	0,167	0,110	0,273	0,159	0,282	0,174	0,360	0,220	0,319	0,210	0,278	-0,040	1,000	0,587	0,569	0,416	0,147
SELF-SCORE-3	-0,522	0,154	0,117	0,111	0,024	0,333	0,243	0,330	0,205	0,397	0,302	0,262	-0,091	0,587	1,000	0,737	0,583	0,190
SELF-SCORE-4	-0,582	0,246	0,199	0,318	0,206	0,303	0,220	0,519	0,389	0,478	0,381	0,393	-0,188	0,569	0,737	1,000	0,853	0,247
SELF-SCORE-5	-0,542	0,293	0,257	0,405	0,305	0,391	0,325	0,539	0,441	0,629	0,541	0,497	-0,251	0,416	0,583	0,853	1,000	0,328
RESPONSE	-0,205	0,472	0,461	0,492	0,483	0,507	0,514	0,432	0,441	0,411	0,429	0,559	-0,107	0,147	0,190	0,247	0,328	1,000

A figura 4.7 mostra a importância de cada atributo tomado de maneira individual e em grupo. A figura 4.7A mostra a AUC dos atributos individuais. Como se pode ver neste gráfico, o melhor atributo simples é "query-score 2"; ou seja, a pontuação da segunda iteração normalizada pela auto-pontuação da *query*. Este atributo, sozinho, alcançou uma AUC de 0.830. Já o melhor atributo (simples ou composto) é a média das pontuações das cinco iterações, normalizadas pela auto-pontuação da *query* e dos *subjects*. Este atributo alcançou uma AUC de 0.878. Lembrando que este atributo é composto, portanto, não é este o "melhor atributo simples" da figura 4.6.

Embora a figura 4.7A nos dê informação sobre a importância dos atributos tomados de maneira individual, ela não nos diz quanto o atributo realmente contribui para o modelo final, uma vez que, a importância real de um atributo depende não só de sua "força" individual, mas também da correlação com outros atributos, conforme já foi dito na introdução (seção 1.1). A figura 4.7B, no entanto, nos dá mais informação a este respeito. Esse gráfico mostra a importância dos atributos medida pela *random forest*. Conforme foi dito na introdução e nos materiais e métodos a RF é capaz de estimar o quanto cada variável contribui para o modelo (seção 3.4.6).

Na figura 4.7B, o que se observa é que os atributos mais importantes são "query-score 3", "subject-score 3" e "average-score". Um fato bastante interessante que se nota na comparação entre 4.7A e 4.7B é que, na primeira, os atributos "query-score" e "subject-score" são semelhan-

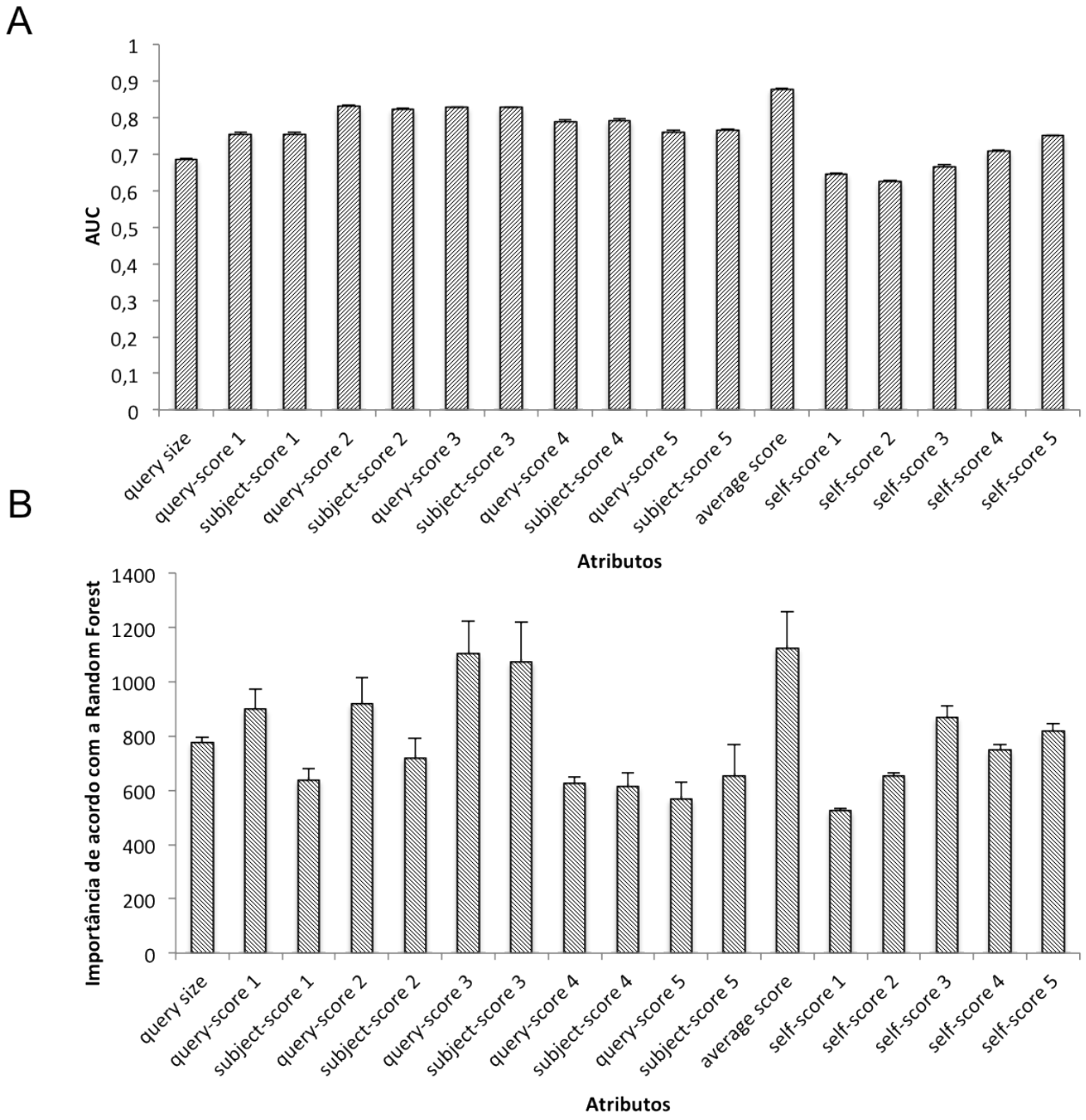

Figura 4.7: Gráficos mostrando a importância de cada atributo. A figura A (acima) mostra a AUC de cada atributo individualmente (incluindo a média dos atributos query-score 1 a subject-score 5). A figura B (abaixo) mostra a importância de cada atributo medido pela *random forest*. As barras de erro representam desvio padrão.

tes, com AUC variando entre 0.753 e 0.830. Enquanto na figura 4.7B, as diferenças são bem mais acentuadas. Isso explica porque estas variáveis são muito correlacionadas entre si.

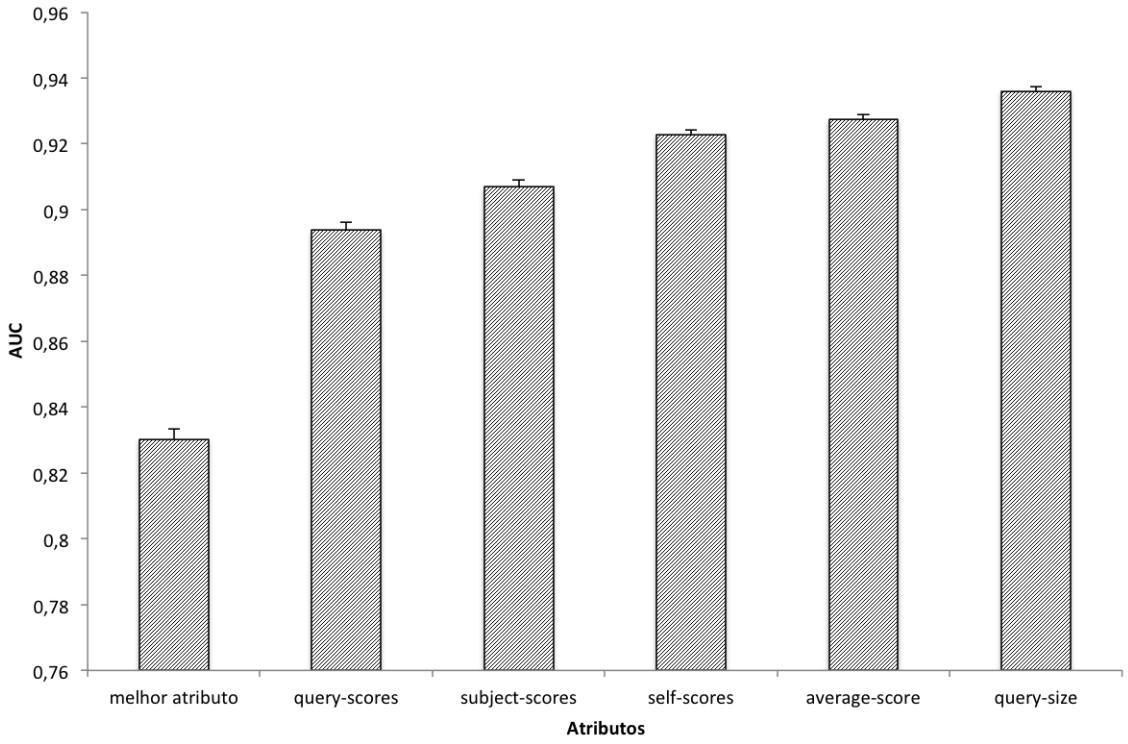

Figura 4.8: Gráfico mostrando a evolução do modelo de *random forest*, com 100 árvores e dois atributos por árvore, à medida que novos atributos são adicionados ao modelo. A barra de "best-feature" representa a AUC do melhor atributo simples e é uma exceção neste gráfico, pois não é um modelo de *random forest*. A barra de "query-scores" representa uma *random forest* treinada com as cinco pontuações normalizadas pela auto-pontuação da *query*. "Subject-score" representa a RF treinada com os cinco atributos anteriores mais as cinco pontuações normalizadas pela auto-pontuação dos *subjects*. "Self-scores" é a RF treinada com os 10 atributos anteriores mais as 5 auto-pontuações. "Average-score" é a RF com os 15 atributos anteriores mais a media dos 10 "query e subjec-scores". Finalmente "query-size" é a RF treinada com os 16 atributos anteriores mais o tamanho da *query*.

A figura 4.7 mostrou a importância de cada atributo, já a figura 4.8 mostra o quanto cada atributo contribuiu para o modelo ao ser adicionado. Nessa figura, observa-se que o maior incremento ocorre ao se construir o modelo com as cinco "query-scores" ao invés de em uma só (a "melhor"). Ao se adicionar mais cinco atributos, os "subject-scores", ocorre um pequeno, porém significativo, ganho ($p<0.05$). O próximo passo foi adicionar os cinco atributos de auto-pontuação. Estes atributos variam apenas de iteração para iteração e de *query* para *query*, mas mesmo assim eles foram capazes de aumentar em 0.016 a AUC. Em seguida adicionamos a média dos 10 primeiros atributos. Este atributo "composto" aumentou a AUC em 0.0045. Este crescimento pode parecer pequeno, mas vale lembrar que foi alcançado adicionando um único atributo, enquanto anteriormente estávamos adicionando grupos de cinco. Finalmente adicionamos o tamanho da *query*, o que aumentou a AUC para 0.936 na *random forest*, um crescimento de 0.0086.

A figura 4.8 foi feita apenas com *random forest*, sem redes neurais. Fizemos assim porque com as RNAs teríamos de reencontrar a quantidade ótima de neurônios na camada oculta toda vez que novos atributos fossem adicionados. A quantidade de neurônios que é ótima para uma quantidade de atributos pode ser demais para um outro número de atributos e levar a *overfiting*. Já com *random forest* o excesso de árvores não leva ao *overfiting*, como na rede neural. Pode ocorrer *overfiting* no sentido de que a predição no treino aumenta mais que no teste, mas não acontece do teste piorar com a RNA (Figuras 4.1 e 4.3).

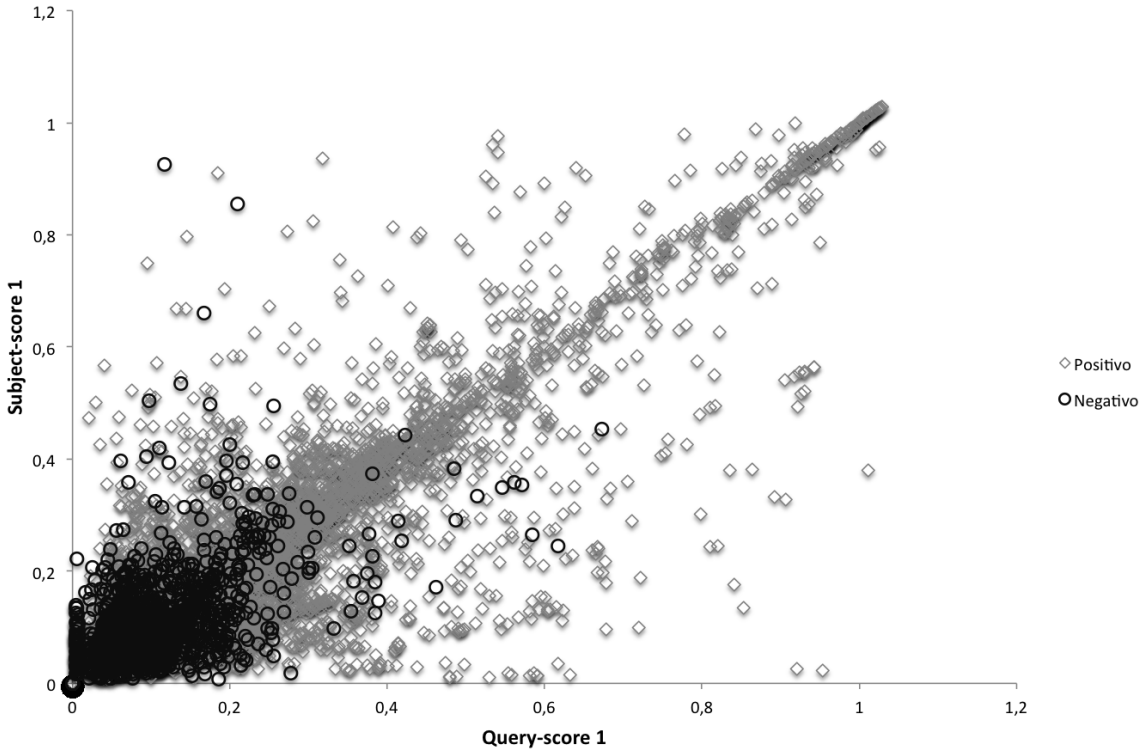

Figura 4.9: Gráfico mostrando a correlação entre 'query-score 1' e 'subject-score 1'. Os losangos cinza representam pontos oriundos de *subjects* do mesmo PANTHER da *query* e os círculos negros representam pontos oriundos de *subjects* de grupos PANTHER diferentes daquele da *query*.

As figuras 4.7 e 4.8 mostraram a importância de cada atributo e como eles contribuíram para o modelo. Finalmente temos a figura 4.9. Esse gráfico mostra a correlação entre 'subject-score 1' e "query-score 1", ou seja, a pontuação da primeira iteração normalizada pela auto-pontuação da *query* e dos *subjects*. A figura mostra um coeficiente de correlação linear de 0.918. Isso mostra que estas variáveis são similares, como previsto na figura 4.7. No entanto, cabe perguntar, por que esta correlação é de 0.918 e não de 1? Em outras palavras, por que estas variáveis não são idênticas uma vez que se trata da mesma pontuação apenas normalizada de duas maneiras diferentes? A resposta é simples e se deve, principalmente, ao tamanho da *query* e dos *subjects*. Suponha uma *query* pequena alinhada com um *subject* grande. Este alinhamento terá uma pontuação, por exemplo, de 90 e a auto-pontuação, oriunda do alinhamento da *query* consigo mesma, será, por exemplo 100. Portanto, o "query-score" será 0.9. No entanto, como o *subject* é grande, sua auto-pontuação, oriunda do alinhamento do *subject* consigo mesmo, será bem

mais alta, por exemplo 1000. Portanto, o ”subject-score” será de 0.09. Com isso nós temos um ’query-score’ auto e um ”subject-score” baixo. Vale lembrar também que dentro de uma iteração a auto-pontuação da *query* será constante, mas cada *subject* terá sua própria auto-pontuação.

Um outro fato que se observa na figura 4.9 é a forma como os pontos ”negativos” e ”positivos” se distribuem. Os primeiros se dispersam próximo da origem enquanto os segundos se concentram ao longo da linha imaginária de 45 graus que vai da origem até a extremidade superior direita do gráfico. De fato foi observado que para os pontos ”positivos” a correlação é de 0.915, enquanto que, para os ”negativos”, esta correlação cai para 0.812.

4.1.2 BLASTp

Semelhante ao que foi feito com PSI-BLAST, será construído a seguir um modelo para validar *hits* de BLASTp. O propósito desta seção não é criar um modelo para competir com modelo do PSI-BLAST, uma vez que, como já foi visto, o BLASTp tem uma taxa de revocação bem mais baixa que o PSI-BLAST. Além do mais, a quantidade de atributos que se pode extrair do BLASTp deve ser menor que do PSI-BLAST. O propósito aqui é criar um modelo heurístico para se evitar fazer PSI-BLAST em situações em que a *query* tenha anotação confiável apenas com BLASTp.

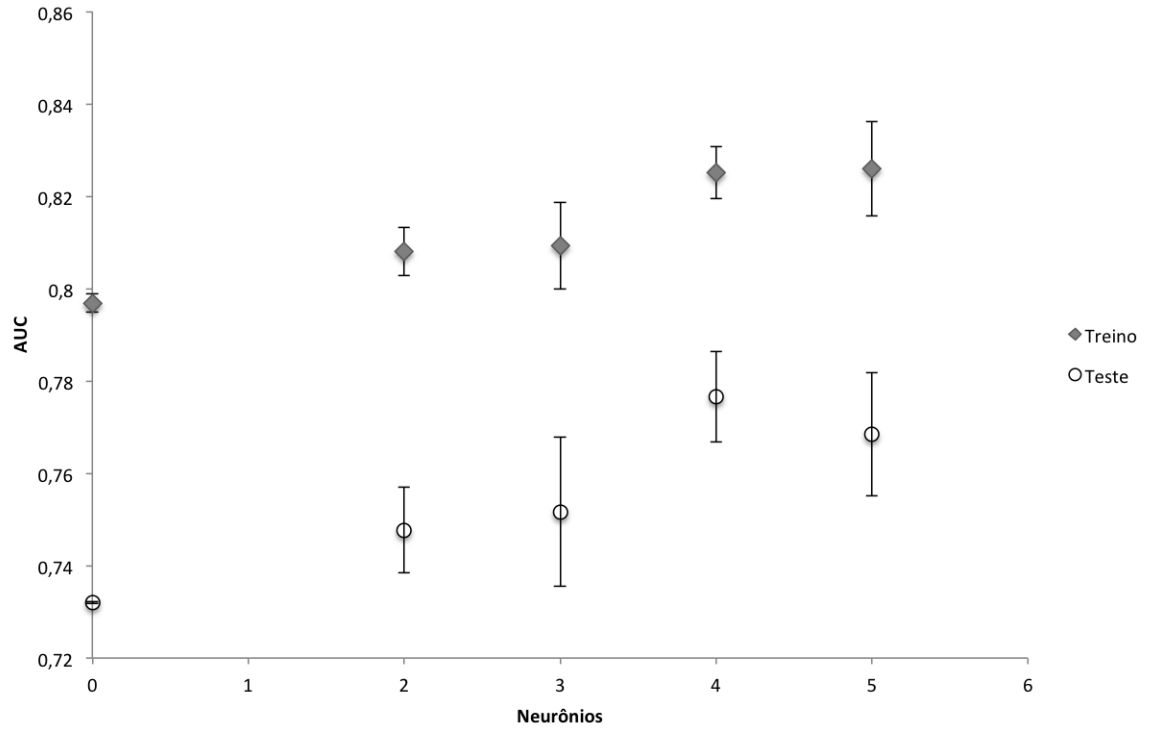

Figura 4.10: Gráfico mostrando a variação da AUC nos dados de treino e de teste com a variação do número de neurônios na camada oculta da rede neural. Cada ponto representa a média de oito réplicas e as barras de erro representam os desvios padrões.

A figura 4.10 mostra a variação do AUC, no treino e no teste, com a variação da quantidade de neurônios na camada oculta. O gráfico mostra que o melhor resultado foi obtido com quatro neurônios (AUC=0.7765). Pode-se ver também nesta figura que a diferença entre treino e teste não cresceu tanto quanto no resultado da figura 4.1. Isso se deve ao fato de que o presente modelo é muito mais simples uma vez que estamos lidando apenas com três atributos, a saber: *"query-score"* pontuação relativizada pela auto pontuação da *query*; *"subject-score"* pontuação relativizada pelas auto pontuações dos *subjects* e *"size"* logaritmo na base dez do número de aminoácidos da *query*.

A figura 4.11 mostra a AUC dos três atributos usados para treinar o modelo bem como o AUC do modelo. O gráfico mostra que a AUC do melhor atributo (*query-score*) é bem próximo da AUC do modelo. O melhor atributo obteve uma AUC de 0.7639 enquanto que o modelo final obteve 0.7765. Esta diferença é pequena, porém, estatisticamente significativa ($p<0.05$).

Assim como no modelo de PSI-BLAST nós também testamos a *random forest*, porém, esta obteve resultado pior que a rede neural e o *ensemble* destes dois modelos não logrou aumentar significativamente a AUC.

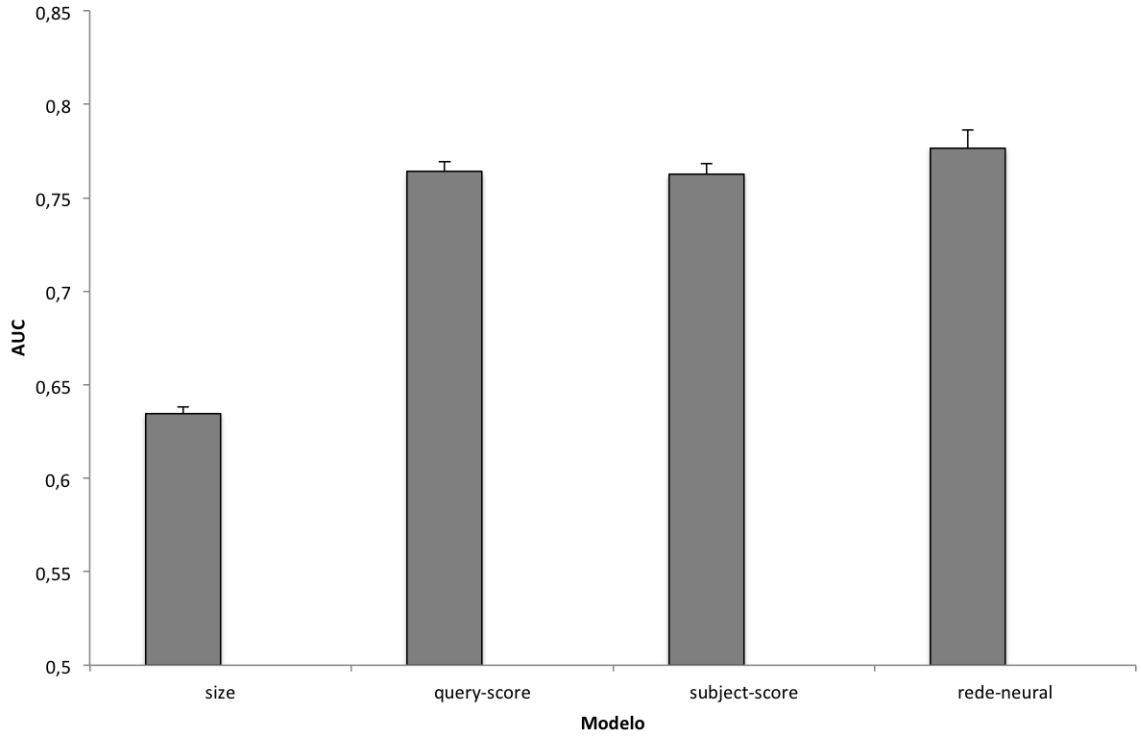

Figura 4.11: Gráfico mostrando a AUC dos atributos individuais e do modelo final. As barras de erro representam desvio padrão.

Ao se comparar a figura 4.11 e 4.7A uma discrepância parece surgir. Os atributos *query-score* e *subject-score* (4.11) possuem AUCs muito diferentes dos atributos equivalentes na figura 4.7A (*query-score-1* e *subject-score-1*). Não obstante, esta discrepância é apenas aparente e se deve ao fato de que no modelo de PSI-BLAST os *subjects* que foram recrutados apenas na segunda interação (ou em uma iteração posterior) receberam pontuação zero na primeira iteração. Isso significa que os atributos *query-score-1* e *subject-score-1* possuem uma série de ”zeros”. Estes ”zeros” em sua maioria são verdadeiros negativos e isso ajuda a aumentar um pouco a AUC do atributo. No caso dos *query-score* e *subject-score* do modelo de BLASTp estas pontuações zero não existem, logo a AUC tende a cair um pouco.

Ao se comparar os resultados obtidos com o modelo de PSI-BLAST e de BLASTp fica evidente a superioridade do primeiro. O primeiro modelo alcançou uma AUC de 0.94 enquanto o segundo apenas 0.7765. O primeiro aumentou a AUC em relação ao melhor atributo individual em 0.11 e o segundo apenas 0.013. Estas observações reforçam a vantagem do modelo baseado em PSI-BLAST, que é o foco deste trabalho. Mas de qualquer forma o modelo baseado em BLASTp servirá como uma heurística para se evitar de fazer PSI-BLAST em alguns casos.

4.1.3 Correlacionando o Modelo com Outras Medidas de Similaridade

Nesta seção iremos mostrar como a confiança calculada pelo modelo desenvolvido na seção anterior se relaciona com outras métricas populares de bioinformática. É claro que estamos falando de métricas populares, mas também de métricas relevantes para o contexto deste trabalho. Para esta série de comparações, 30 proteínas contento PDB foram submetidas aos nossos procedimentos (Materiais e Métodos 3.4.7). Escolhemos proteínas com PDB porque uma das comparações que faremos envolve estrutura secundária.

Iniciaremos esta série de comparações com a métrica mais simples e trivial, a identidade calculada pelo BLAST. Na figura 4.12 pode-se ver o gráfico da identidade, entre o *subject* e a *query*, contra a confiança calculada pelo nosso modelo. Nesse gráfico pode-se ver que a identidade tende a cair à medida que confiança diminui. O gráfico também mostra que há uma grande dispersão nos dados.

A identidade é uma métrica simples e não é possível, apenas com ela, definir um grupo ou família de proteínas tal como o PANTHER, KO ou Pfam entre outros. Dessa forma, um modelo capaz de recrutar proteínas da mesma "família", ou que possuam função similar, deve ser capaz de ir além da identidade. E é exatamente isso que nosso modelo faz. Ele gera uma confiança que tem certa correlação com a identidade (r=0.68), mas também é capaz de agrupar (alta confiança) proteínas com identidade muito baixa. Além do mais, se a correlação com a identidade fosse total (r=1) não haveria necessidade de um modelo sofisticado uma vez que bastaria usar a identidade.

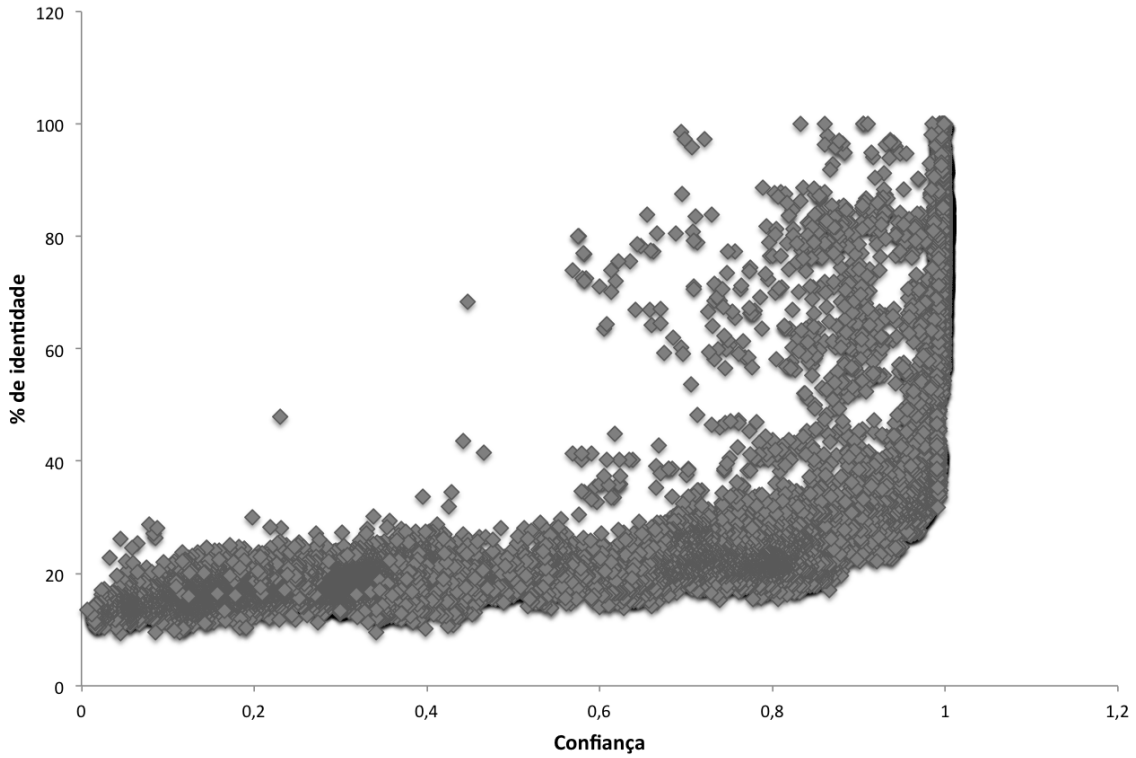

Figura 4.12: Gráfico mostrando a dispersão da porcentagem de identidade calculada pelo BLAST com a confiança calculada pelo nosso modelo.

Em seguida testamos a "correlação" da confiança com a distância filogenética. Esta métrica é similar à identidade, no sentido que ela também é uma forma de medir "similaridade" das estruturas primárias das proteínas. No entanto, esta métrica é bem mais sofisticada que a primeira por fazer uso de um alinhamento global (ver introdução seção 1.4.4). É importante lembrar também que a distância pontua mais, quem é mais diferente. Essa é uma propriedade intrínseca de distâncias.

A figura 4.13 mostra o gráfico da distância filogenética contra a confiança. Este gráfico mostra que, em média, quanto maior a confiança menor a distância. O que se observa neste gráfico é que ele é bem mais linear que o primeiro, mas, assim como o anterior, também há uma grande dispersão dos dados. Foi medido neste gráfico um coeficiente de correlação linear de 0.87. Esta maior correlação provavelmente se deve mais à maior linearidade dos dados do que a uma menor dispersão. Novamente nosso modelo se correlaciona com esta medida, mas a informação que ele retorna está longe de ser a mesma retornada pela distância filogenética.

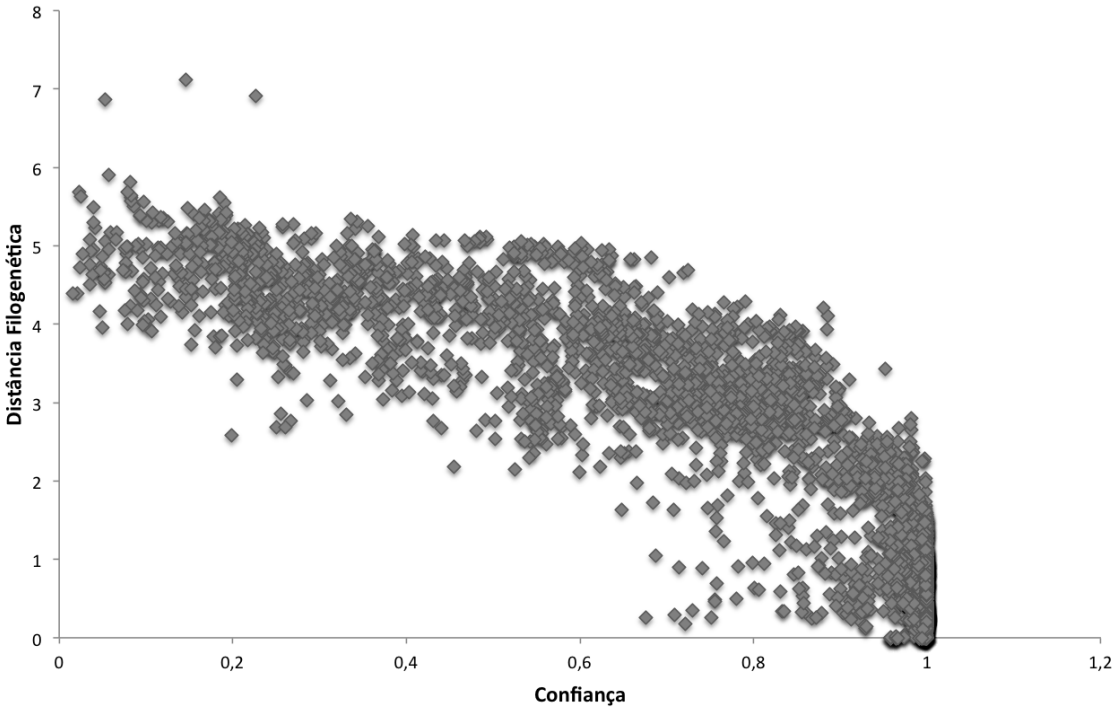

Figura 4.13: Gráfico mostrando a dispersão da distância filogenética entre as *queries* e os *subjects* com a confiança calculado pelo nosso modelo.

Como foi dito na introdução, o PSI-BLAST é uma ferramenta capaz de recrutar proteínas que tenham similaridade baixa, em termos de estrutura primária, mas que sejam muito parecidas na estrutura secundária. Algumas ferramentas de predição de estrutura secundária, inclusive, usam PSI-BLAST. Nossa próxima comparação lida exatamente com isso. Como já foi dito, todas as *queries* usadas nos testes desta seção possuem PDB. Algumas das proteínas que estas *queries* recrutam também possuem PDB e, desta forma, suas estruturas secundárias podem ser comparadas com as estruturas das *queries*. Para se medir a semelhança entre estas estruturas foi usada a métrica SOV. Conforme foi dito na introdução, esta métrica foi desenvolvida primariamente

para se comparar estrutura secundária real com estrutura secundária predita de uma mesma proteína. No entanto, em nosso laboratório, o SOV foi "adaptado" para se comparar estrutura de proteínas diferentes mediante o alinhamento das sequências de aminoácidos (ver materiais métodos seção 3.3.6).

A figura 4.14 mostra a distribuição de SOV por faixa de confiança. Diferentemente das figuras anteriores, não existe muita correlação entre SOV e confiança. Todavia, o que se observa no gráfico é que nosso modelo recruta quase que exclusivamente proteínas com estruturas secundárias bem similares. No gráfico 4.14 pode-se identificar três grupos levemente diferentes entre si. Primeiro, temos as proteínas com confiança maior que 90% (caixa mais à direita no gráfico). Essas sequências têm SOV mediano de 95%. O segundo grupo são aquelas sequências com confiança entre 50-90% (caixas 2-5 da direita para a esquerda) que possuem SOV mediano médio de 81% e uma dispersão alta, conforme se observa pela largura dos quartis e pelos valores extremos. Entretanto, a maioria das proteínas neste grupo tem SOV acima de 60%. Por fim, o terceiro grupo é o das proteínas com confiança menor que 50% (5 caixas mais à esquerda no gráfico). Esse grupo tem SOV mediano médio de 75% e uma dispersão baixa. De fato, a maior parte das proteínas do terceiro grupo tem SOV acima de 70%. Assim, o que esta figura mostra é que mesmo aquelas proteínas com confiança baixa possuem uma similaridade alta em termos da estrutura secundária.

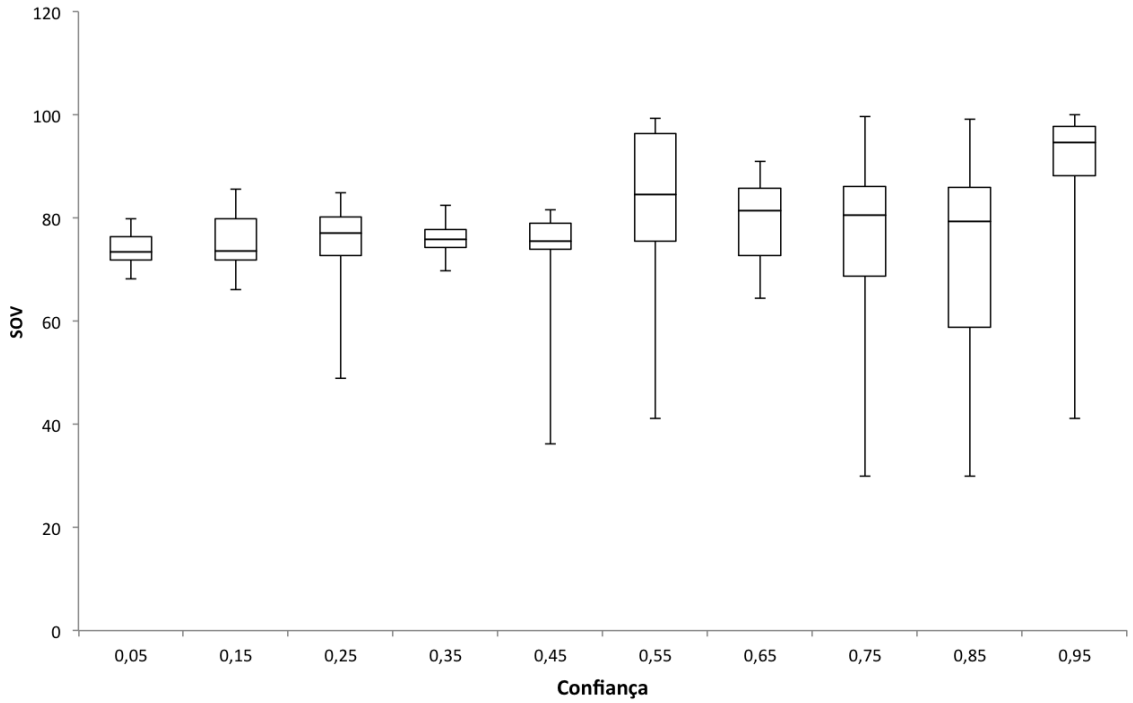

Figura 4.14: *Boxplot* mostrando a variação do SOV entre as *queries* e os *subjects* com a confiança calculada pelo nosso modelo. As linhas no interior das caixas representam as medianas, as bordas superior e inferior das caixas representam o primeiro e terceiro quartis respectivamente e as barras de erro representam os valores máximos e mínimos.

A figura 4.15 mostra a frequência relativa de SOV. Nesta figura, observa-se que apenas 1% das proteínas recrutadas pelo nosso modelo possuem SOV menor que 40% e que 58% tem SOV

entre 80-100%. Este resultado confirma que nosso modelo recruta quase que exclusivamente proteínas parecidas em termos de estrutura secundária

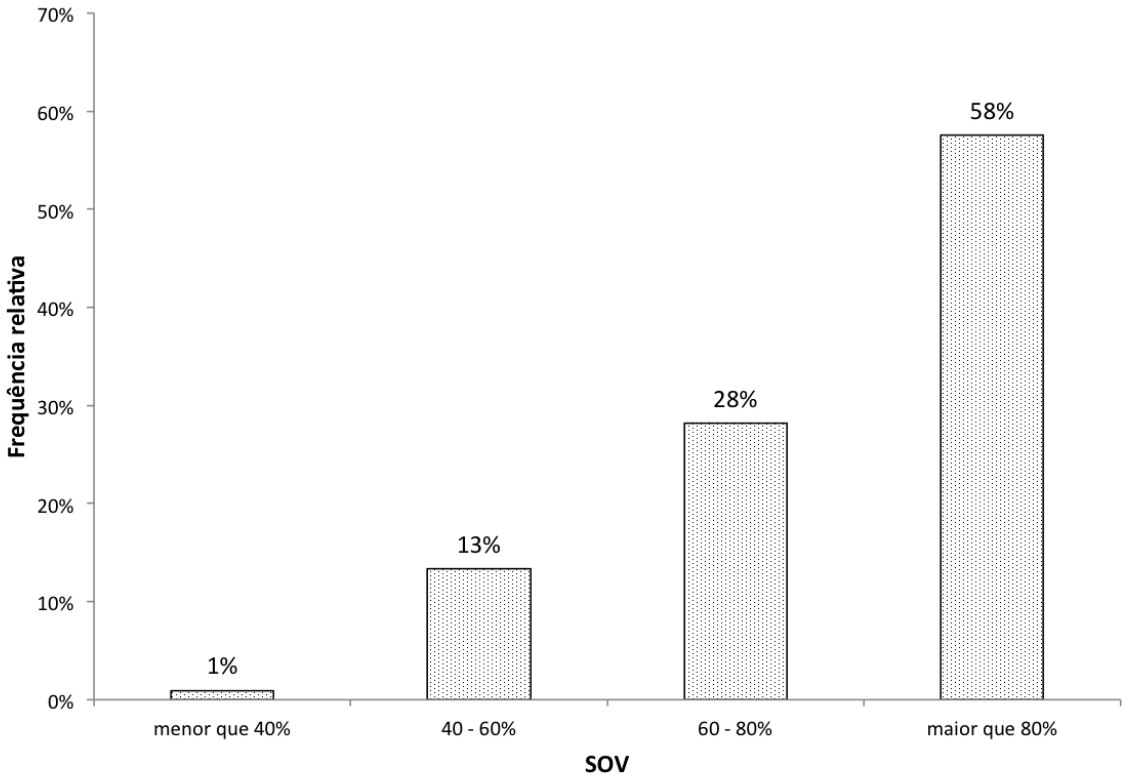

Figura 4.15: Gráfico mostrando a frequência relativa dos SOVs medidos entre as *queries* e os *subjects*. As barras representam quantos por cento dos SOVs medidos estão abaixo de 40%, entre 40-60%, entre 60-80% e entre 80-100% respectivamente. Os números acima das barras representam a altura (frequência) exata da barra.

A última comparação que fizemos foi entre confiança e LCA. Essa técnica diz qual o ancestral comum entre dois organismos. No caso deste trabalho, o ancestral comum entre o organismo da *query* e do *subject*. Na verdade, o que vamos medir aqui é qual o menor nível da árvore filogenética que os dois organismos compartilham. Esta árvore filogenética possui 18 níveis onde 18 significa mesmo genoma, 17 mesma subespécie, 16 mesma espécie, 15 mesmo gênero e assim por diante. Na outra ponta temos dois significando mesmo reino, um mesmo super-reino e zero, *cellular-organisms*.

A figura 4.16 mostra o LCA médio por faixa de confiança. Como se pode ver nesta figura, proteínas com confiança acima de 0.9 têm LCA médio 5.3 e proteínas com confiança abaixo de 0.1 tem LCA médio de 1.0. Claro que isso é uma média. Se plotássemos os dados ”brutos” (não agrupados) observaríamos que em qualquer faixa de confiança nosso modelo recruta proteínas com LCA variando de 0 a 18. No entanto, como se pode ver na figura 4.16 a medida que a confiança cai, o modelo tende a pegar cada vez mais proteínas com LCA baixo e menos proteínas com LCA auto.

Em síntese, estes resultados mostram que nosso modelo recruta, progressivamente, proteí-

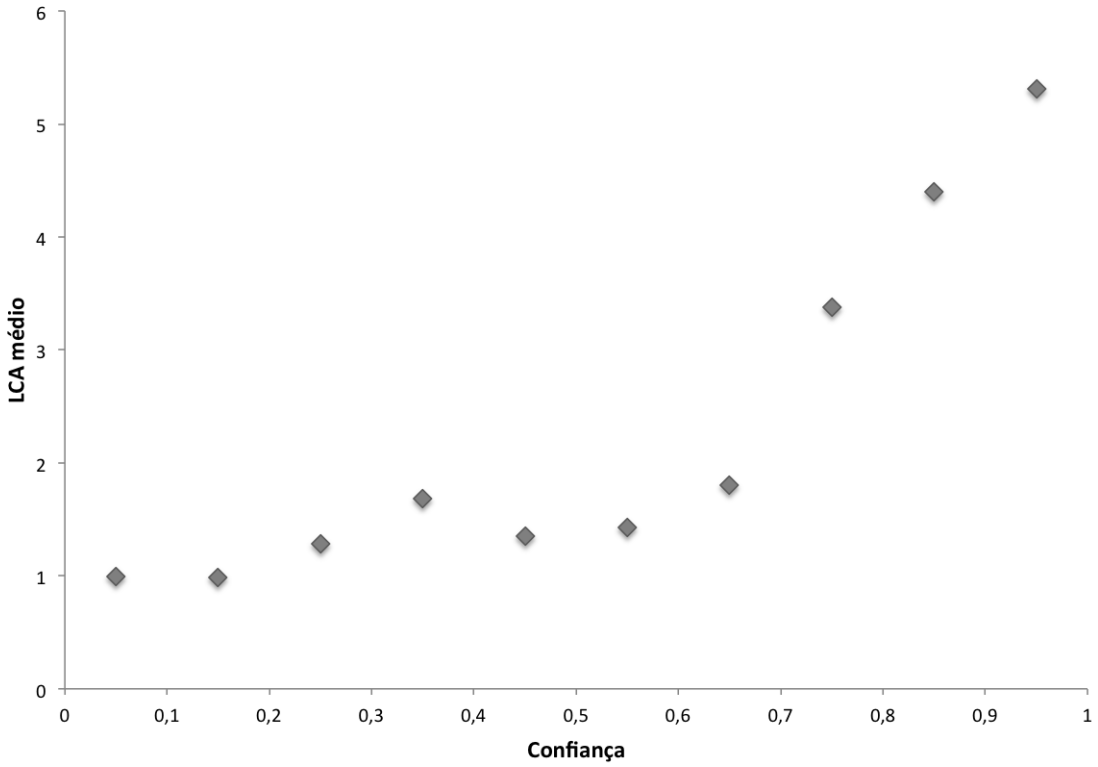

Figura 4.16: Gráfico mostrando o LCA médio, entre as *queries* e os *subjects*, por faixa de confiança calculada pelo nosso modelo.

nas com similaridade cada vez mais distante. A similaridade medida tanto pela sequência de aminoácidos quanto pelo ”grau de parentesco” entre os organismos de origem das proteínas (*query* e *subject*). No entanto, o modelo garante alta similaridade no que diz respeito a estrutura secundária, não importando a confiança.

4.2 Mineração de Anotações

Na seção anterior mostramos o procedimento para validação dos resultados de BLASTp/PSI-BLAST. Desenvolvemos uma metodologia capaz de ordenar os resultados do BLAST do mais provável para o menos provável de pertencer ao grupo da *query*. No entanto, neste trabalho estamos desenvolvendo um procedimento para sugestão de anotação de proteínas hipotéticas. Assim sendo, nesta seção mostraremos um método para mineração de anotações proteicas a partir das anotações encontradas pelo modelo descrito anteriormente.

Tabela 4.2: Tabela mostrando sete resultados sorteados da proteína A0AIS1. A primeira coluna contem o ID da *query*, a segunda coluna contem o ID dos *subjects*, a terceira coluna a confiança do *subject*, a quarta coluna contem a descrição da *query* e a quinta coluna a descrição dos *subjects*.

Query	*Subject*	Confiança	Anotação da *Query*	Anotação do *Subject*
A0AIS1	D3UN55	0.9994	Ribosomal RNA small subunit methyltransferase	Diacylglycerol kinase domain protein
A0AIS1	F3RJY6	0.9993	Ribosomal RNA small subunit methyltransferase	6S ribosomal RNA methyltransferase RsmE
A0AIS1	G5JLT0	0.9014	Ribosomal RNA small subunit methyltransferase	16S ribosomal RNA methyltransferase RsmE
A0AIS1	G6KLA3	0.8142	Ribosomal RNA small subunit methyltransferase	Ribosomal RNA small subunit methyltransferase E
A0AIS1	Q1J4K1	0.8008	Ribosomal RNA small subunit methyltransferase	cytosolic protein
A0AIS1	E6HNQ4	0.6123	Ribosomal RNA small subunit methyltransferase	RNA methyltransferase, RsmE
A0AIS1	E0GD33	0.5361	Ribosomal RNA small subunit methyltransferase	RNA methyltransferase, RsmE

A tabela 4.2 mostra algumas instâncias sorteadas da aplicação do nosso modelo à proteína A0AIS1 (Materiais e Métodos 3.4). Nesta tabela, podemos ver os ids da *query* e do *subject*, a confiança calculada pelo modelo descrito na seção anterior e as anotações da *query* e do *subject*. Observa-se que a tabela está ordenada pela confiança e que a *query* tem função conhecida (*16S ribosomal RNA methyltransferase RsmE*). Finalmente, um fato importante, mas já esperado, é que as funções dos *subjects* nem sempre concordam com a função da *query*. O problema então é encontrar a anotação mais provável.

Uma solução comum para este problema é o uso do melhor resultado (*best-hit*). Uma segunda solução é usar a anotação mais popular. No entanto, duas anotações podem ser parecidas mas não exatamente iguais, dificultando, pelo menos a princípio, definir o que seria uma "anotação mais popular". A maneira mais simples de resolver este problema é contar as palavras que aparecem nas anotações e, então, "pontuar" cada anotação conforme a popularidade de suas palavras. Por exemplo, na tabela 4.2 as palavras que aparecem mais são "*methiltransferase*" e "RNA" (5 vezes cada), portanto, as anotações que contiverem estes termos devem receber mais "pontos". De fato, a anotação real, que neste caso é conhecida, contém estas palavras. Não obstante, considerar as anotações com confiança alta com o mesmo peso daquelas com confiança baixa, uma vez que, estas últimas, provavelmente, nem estão corretas, pare ser um procedimento sub-ótimo. Neste trabalho desenvolvemos um mecanismo para fazer este tipo

de ponderação e então reordenar as anotações das mais confiáveis para as menos confiáveis.

No parágrafo anterior, apresentamos em altíssimo nível o nosso raciocínio de mineração de anotações usando uma linguagem relativamente informal, mas, conforme foi dito nos materiais e métodos (3.6), nós criamos um vetor com o consenso das anotações e ordenamos cada anotação então pelo cosseno do ângulo do vetor anotação com o vetor consenso. Onde o vetor anotação é um vetor de *bag of words* e o vetor consenso é um vetor com a contagem das palavras, podendo esta contagem ser ponderada ou não. Quando dissemos contagem queremos dizer em quantas anotações a palavra ocorreu.

Como dissemos no parágrafo anterior a contagem do vetor consenso pode ser ponderada e é neste ponto que entra a figura 4.17. Estes gráficos mostram a "similaridade" média entre as anotações dos *subjects* e das *queries* à medida que se varia a confiança. A figura 4.17A mostra a similaridade do cosseno média entre as anotações. Já na figura 4.17B, o que se vê é a porcentagem com que um termo da anotação do *subject* esta presente na anotação da *query*. Comparando-se os gráficos A e B observa-se que as escalas são diferentes mas que os perfis são quase idênticos (correlação de 0.9907). Analisando a figura 4.17B, que possui uma interpretação mais direta por se tratar de porcentagem, observa-se que com confiança de 99% (0.99, eixo X) a porcentagem (eixo Y) de um termo individual da anotação estar correto é de 60% e com confiança de 1% (0.01, eixo X) esta porcentagem (eixo Y) cai para 1.3%. Estes números nos dizem o quanto devemos "acreditar" nas anotações dependendo da confiança que nosso modelo calculou e são estas curvas que serão usados para se ponderar as contagens do vetor consenso.

Em ambas as curvas da figura 4.17, fomos capazes de ajustar um polinômio de terceiro grau. Embora estes polinômios sejam levemente diferentes eles possuem correlação de 0.9907 e para os procedimentos que estamos adotando qualquer forma de ponderação que tenha correlação de 1.0 (ou muito próximo) é idêntica. No entanto, a partir deste ponto usaremos o cosseno (figura 4.17A) por ter apresentado menos ruído nos dados "brutos" (não agrupados).

A figura 4.18 mostra a correlação da similaridade do cosseno, entre as anotações da *query* e dos *subjects*, com as métricas identidade, pontuação do PSI-BLAST e confiança calculada pelo nosso modelo. Estas correlações não podem ser comparadas com o quadrado da correlação da figura 4.17 porque, naquela figura, os dados foram agrupados em classes de confiança e o r-quadrado era entre os dados reais e o polinômio ajustado. Já na figura 4.18 temos a correlação linear entre as métricas apresentadas, incluindo porcentagem e similaridade de cosseno.

O fato importante que se vê na figura 4.18 é que a confiança calculada por nosso modelo tem uma correlação com as similaridades das anotações bem maior que as outras métricas, que são oriundas de PSI-BLAST puro. Isso mais uma vez reforça o poder e a importância do nosso modelo.

Para se obter as figuras 4.17 e 4.18 foram feitos alguns pré-processamentos simples, típicos de mineração de texto, como remoção de palavras pouco informativas (*stopwords*), entre outros. A figura 4.19 mostra os principais passos da evolução do modelo de mineração de anotações. Nessa figura, a correlação do cosseno, entre as anotações da *query* e dos *subjects*, com a confiança foi medida após vários pré-processamentos serem feitos. Sendo que a barra "simples" representa o modelo construído usando cada palavra individualmente (*singleGram*) e sem remoção de *stopWords*. A barra "*digram*" representa o modelo feito com palavras tomadas de maneira

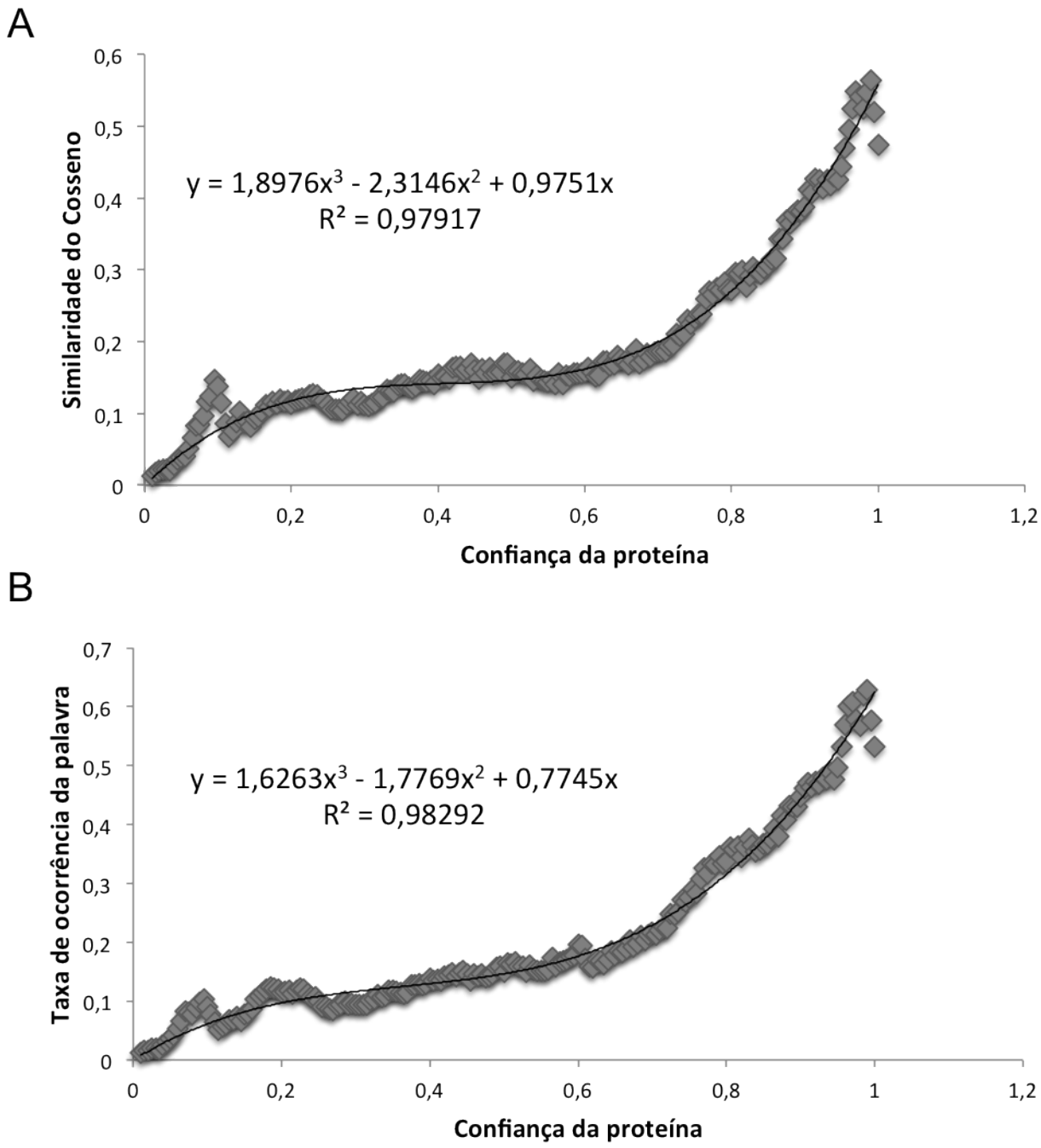

Figura 4.17: Gráficos mostrando a variação média da similaridade da anotação da *query* e do *subject* à medida que se varia a confiança do *subject*. O gráfico A mostra a similaridade do cosseno média entre as anotações e o gráfico B mostra a porcentagem com que uma palavra da anotação do *subject* esta presente na anotação da *query*. Os gráficos também mostram o polinômio de terceiro grau que foi ajustado a cada curva e o r-quadrado destes ajustamentos. Estes dois polinômios por sua vez tiveram uma correlação de 0.9907 entre si na faixa apresentada nos gráficos.

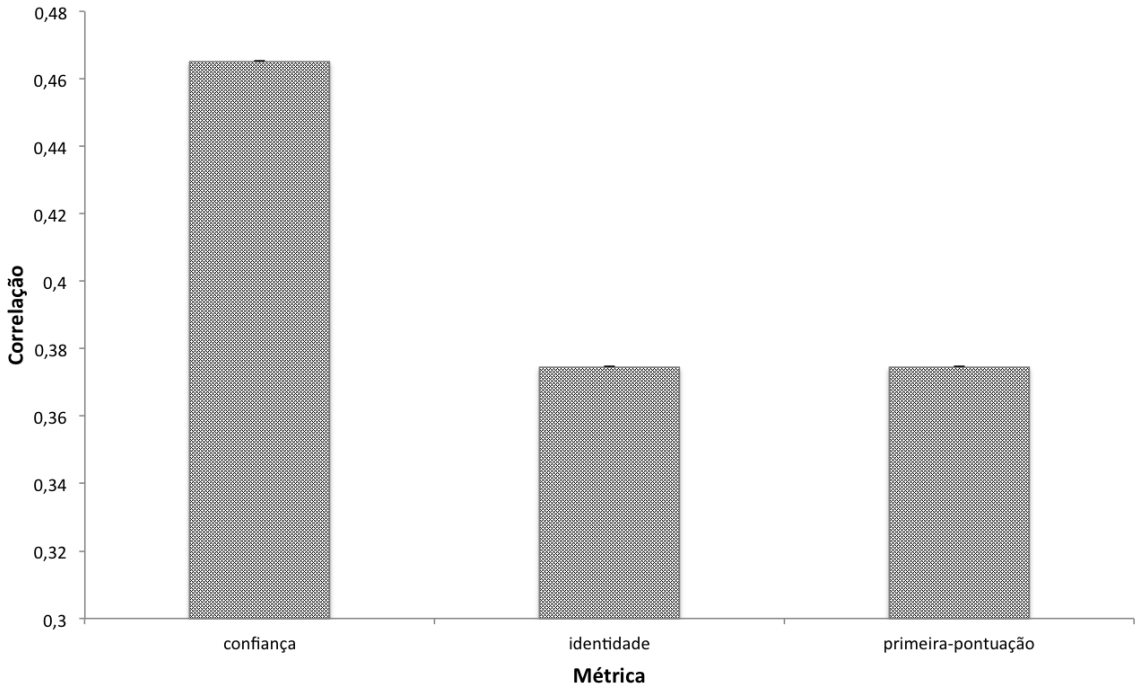

Figura 4.18: Gráficos mostrando a correlação da similaridade do cosseno, entre anotação da *query* e do *subject*, e a pontuação, a identidade e a confiança calculada pelo nosso modelo. As barras de erro representam desvio padrão.

individual e agrupadas duas a duas mas ainda sem remoção de *stopWords*. As barras seguintes também foram feitas com *digrams*, porem diferentes listas de *stopWords* foram testadas. Na barra MySQL usamos a lista de *stopWords* do MySQL; na barra *bioStopWords* usamos uma lista que nós criamos e que pode ser vista no apêndice A. A barra *bioStopWords*-expandidas também usamos a lista que criamos, mas acrescentamos a ela palavras que têm "alta" taxa de co-ocorrência com o termo *"uncharacterized"*. E, na barra "baixa contagem", acrescentamos ainda, palavras que ocorreram 10 ou menos vezes em todo o UniProt.

Como se pode ver nessa figura, o uso de *digrams* levou a uma pequena melhoria na correlação. Em seguida, testamos várias listas de *stopWords* e como se observa a lista do MySQL não melhorou o resultado, no entanto, a lista por nós definida fez a correlação crescer enormemente. A expansão desta lista com as palavras que co-ocorrem com *"uncharacterized"* também conseguiu aumentar a correlação um pouco mais. Por fim, acrescentamos palavras com baixa contagem (menos de 11 no UniProt), mas esta não levou a melhoria neste teste. No entanto, manteremos esta lista, pois ela remove ruídos como palavras escrito erradas, como *"hipothetical"*, bem como a redução da dimensionalidade do problema. Além do que mostramos no gráfico, também testamos *stemming*, no entanto, esta técnica não logrou aumentar a correlação

Conforme foi dito as *bioStopWords* foram definidas por nós neste trabalho e a lista completa pode ser encontrada no apêndice A. Para a criação desta lista de 113 palavras, nós fizemos uma contagem das palavras que ocorreram no UniProt (versão de fevereiro de 2012) e selecionamos aqueles termos que apareceram mais de 10 mil vezes. Isso retornou 672 palavras. A partir dessas palavras selecionamos manualmente aquelas que eram pouco informativas. Como se

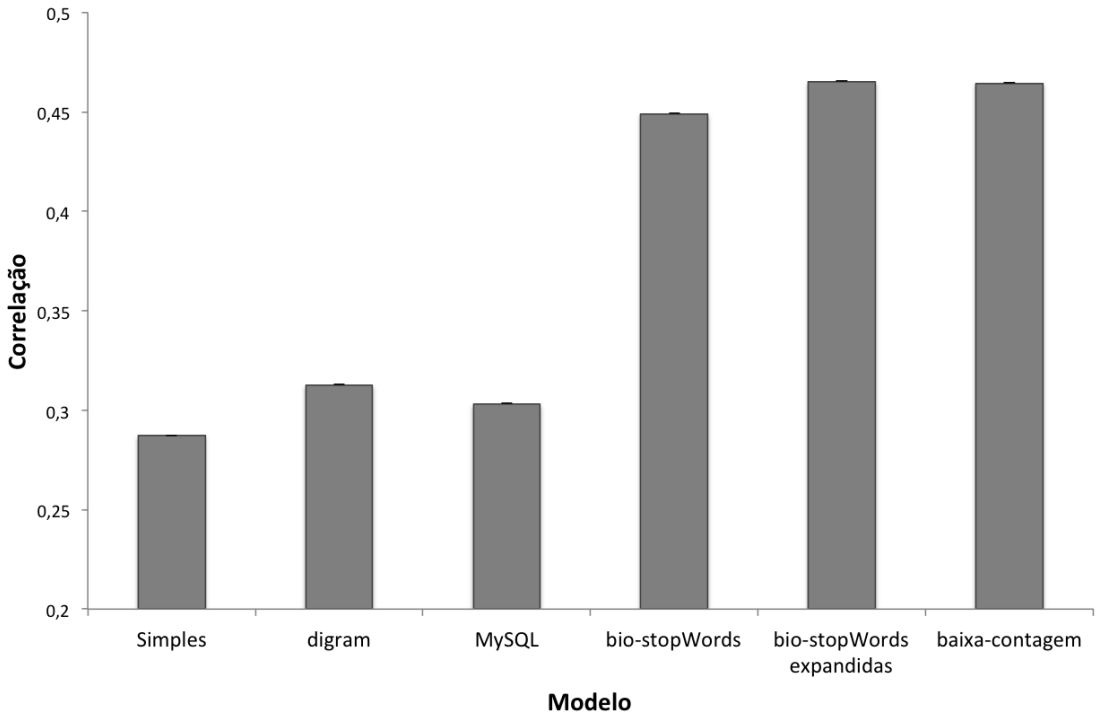

Figura 4.19: Gráfico mostrando a evolução do modelo de mineração de anotações. A barra 'simples' representa modelo construído com palavras individualmente e sem exclusão de *stopWords*. Na barra '*digram*' as palavras foram tomadas duas a duas e individualmente mas sem exclusão de *stopWords*. Nas demais barras, as palavras também foram tomadas individualmente e duas a duas mas diferentes listas de *stopWords* foram usadas. Na barra 'MySQL' foi usada a lista de *stopWords* do MySQL, na barra '*bio-stopWords*' foi usada a lista *stopWords* que nós criamos, na barra '*bio-stopWords*-expandidas' foi usada a lista de *stopWords* que nós definimos acrescida de termos que co-ocorrem com a palavra *uncharacterized* e na barra 'baixa-contagem' foi usada a lista anterior mais o conjunto das palavras que apareceram menos de 10 vezes em todo o UniProt. As barras de erro representam desvio padrão.

pode ver, a lista (apêndice A) inclui termos de ligação como *"for"* e *"to"*, termos de anotação desconhecida como *"uncharacterized"* e *"hypothetical"*, além de outras palavras encontradas na anotação de um amplo espectro de proteínas que não possuem nenhuma similaridade entre si, sendo que algumas dessas são bem retóricas, como *"protein"*, uma vez que estamos lidando com banco de dados de proteínas e "toda proteína é uma proteína".

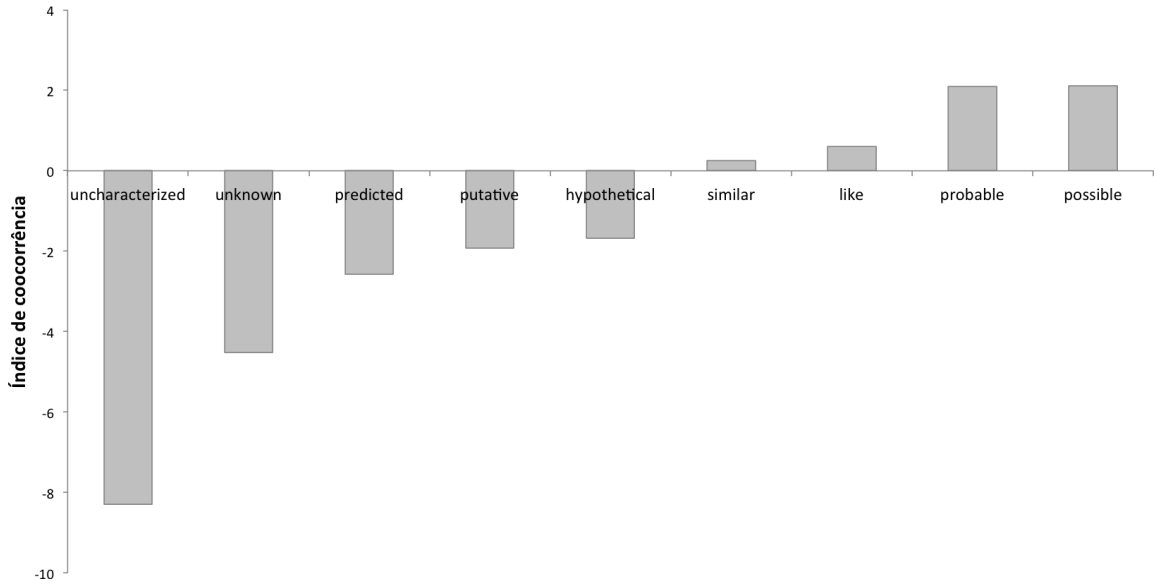

Figura 4.20: Gráficos mostrando o índice de co-ocorrência entre palavras terminadas em "ase" e nove palavras selecionadas características de anotação hipotética.

Nós dissemos que na barra de *bioStopWords*-expandidas 4.19 foram acrescentadas palavras que co-ocorreram com o termo *"uncharacterized"*. Para se medir o índice de co-ocorrência, usamos a equação 3.2. Esta equação pontua com valor positivo, termos que co-ocorrem com frequência acima do esperado por mero acaso; negativamente palavras que co-ocorrem menos do que o esperado por acaso e zero se a chance é a mesma esperada por mero acaso. Usando, então, esta equação adicionamos todas as palavras que tiveram índice de co-ocorrência com *"uncharacterized"* maior ou igual a zero. É claro que, em geral, zero seria um valor baixo, pois zero representa co-ocorrência por mero acaso. No entanto *"uncharacterized"* é uma palavra "forte" em termos de função desconhecida. Uma palavra não co-ocorre com '*uncharacterized*' se ela indicar função da proteína.

A figura 4.20 mostra a co-ocorrência de nove palavras indicadoras de proteína hipotética e função desconhecida com palavras terminadas em "ase". Essas palavras, com raríssimas exceções, são nomes de enzimas e as proteínas anotadas com essas palavras quase sempre são enzimas, proteínas semelhantes a enzimas ou proteínas relacionadas a enzimas, logo, são proteínas com funções relativamente bem definidas. Como se pode ver nessa figura, o termo *"uncharacterized"* é o que tem a taxa de co-ocorrência mais baixa confirmando o que foi conjeturado no parágrafo anterior. De fato, uma contagem rápida mostra que esta taxa é cerca de 250 vezes menor do que o esperado aleatoriamente. Além do que o gráfico mostra, uma checagem manual mostrou que apenas sete palavras terminadas em 'ase' tem índice de co-ocorrência maior que zero com "*uncharacterized*". Sendo que destas, cinco são nomes de

enzimas com erro de escrita e nenhuma delas apareceu mais que três vezes em todo o UniProt. As outras duas são as palavras "mase" e "vase", duas das raras exceções que dissemos que poderiam ocorrer, mas, novamente, estas palavras não apareceram mais que 60 vezes em todo o UniProt. Já no outro extremo dos resultados do gráfico 4.20, temos os termos *like*, *probable* e *possible* que tendem a co-ocorrer com anotações de enzimas, o que é mais ou menos evidente, já que estas palavras aparecem em contextos como "possível 'alguma função"', "provável 'alguma função"' e assim por diante.

Os resultados apresentados anteriormente mostraram como podemos ponderar um vetor consenso. Agora iremos mostrar como a ordenação das anotações por sua similaridade com este vetor consenso contribui para a sugestão de anotação de proteínas hipotéticas. A Figura 4.21 mostra a distribuição das similaridades do cosseno entre a melhor sugestão de anotação e a anotação real da *query*. Nessa figura, três formas distintas de "melhor sugestão de anotação" foram testadas. Na primeira, chamada "polinomial", um vetor consenso foi criado, usando os procedimentos descritos acima (gráficos 4.17) e nos materiais e métodos (seção 3.6.5 e equações 3.4 e 3.6), e a anotação com maior similaridade com o vetor consenso foi retornada. A segunda, chamada "constante", é bem semelhante à primeira exceto que o vetor consenso foi obtido sem nenhuma ponderação (Materiais e Métodos seção 3.6.5 e equações 3.3 e 3.7). A terceira, chamada "melhor resultado", Simplesmente, usamos a anotação do *subject* com confiança mais alta (ver seção 3.6.10). É claro que as anotações que possuem apenas *stopWord* foram ignoradas.

O gráfico 4.21A mostra que o nosso procedimento de mineração retornou 43% de anotações com similaridade do cosseno, em relação a anotação real, maior que 0.8 e 23% de anotação com cosseno menor que 0.2. Já sem a ponderação, a taxa de cosseno maior que 0.8 cai para 39% e a de cosseno menor que 0.2 sobe para 42%. E para o *best-hit*, a taxa de cosseno maior que 0.8 é de apenas 29% e a de cosseno menor que 0.2 é de 47%. A figura 4.21B mostra a similaridade do cosseno média entre a anotação do melhor *subject* e a anotação da *query*. Essa figura confirma, como ficou patente no gráfico 4.21A, que nosso modelo de mineração de dados traz melhoria para o modelo final.

O procedimento descrito anteriormente se aplica ao modelo de PSI-BLAST, mas nós também desenvolvemos um modelo heurístico para BLASTp. Isso significa que esse modelo deve ter seu próprio procedimento de mineração de anotação. Evidentemente não será preciso redefinir lista de *stopWords* ou estratégias de pré-processamento de texto, mas a curva da figura 4.17 precisa ser checada. Embora não tenha sido mostrada, a similaridade entre anotações dos *subjects* e das *queries* cai de maneira aproximadamente linear com a confiança do *subject*. No entanto, ponderar o consenso por essa reta não trouxe melhoria estatisticamente significativa como será mostrado a seguir.

A figura 4.22 mostra um procedimento bem semelhante àquele mostrado na figura 4.21, exceto que esta figura foi feita usando confiança do modelo de BLASTp (quando aplicável) e, ao invés de "polinomial", nós temos "linear". O que esse gráfico mostra é que usar a anotação que mais concorda com o consenso logrou melhor resultado que o *best-hit*. No entanto, a ponderação linear obteve quase que exatamente o mesmo resultado do consenso não ponderado. Isso pode ser observado tanto na distribuição 4.22A quanto na média 4.22B.

Ao se comparar os resultados das figuras 4.21 e 4.22 fica evidente que a mineração usando

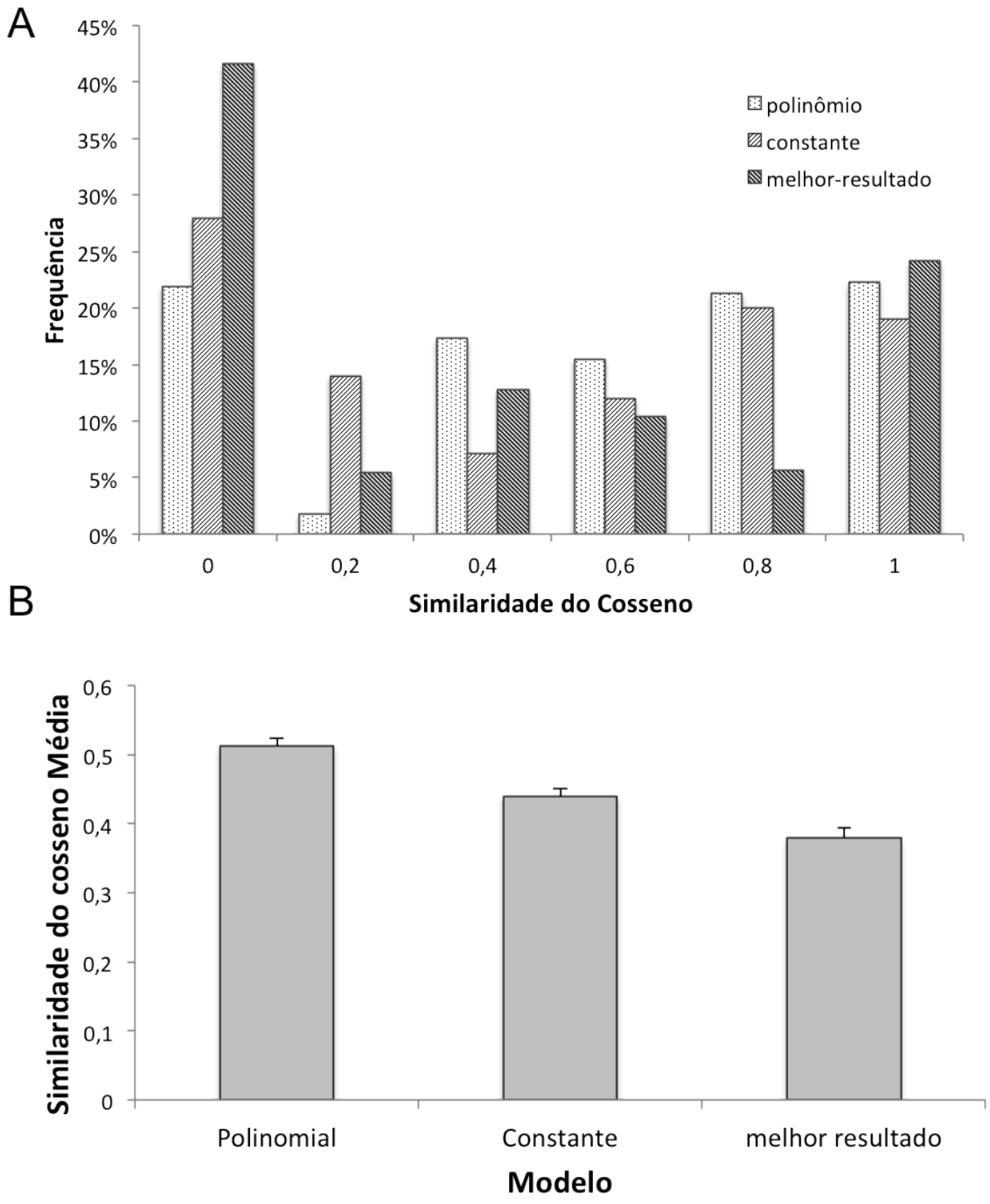

Figura 4.21: Gráficos mostrando similaridade do cosseno entre as anotações das *queries* e dos *subjects* selecionados por três métodos distintos a partir do modelo de PSI-BLAST. A figura 'A' mostra a distribuição da similaridade do cosseno quando os *subjects* são selecionados pelo método de ponderação polinomial (polinomial), sem ponderação (constante) e seleção do *subject* com maior confiança (melhor resultado). A figura B mostra a similaridade do cosseno médio entre as *queries* e os *subjcts* selecionados da forma já descrita. As barras de erro, no gráfico B, representam erro padrão.

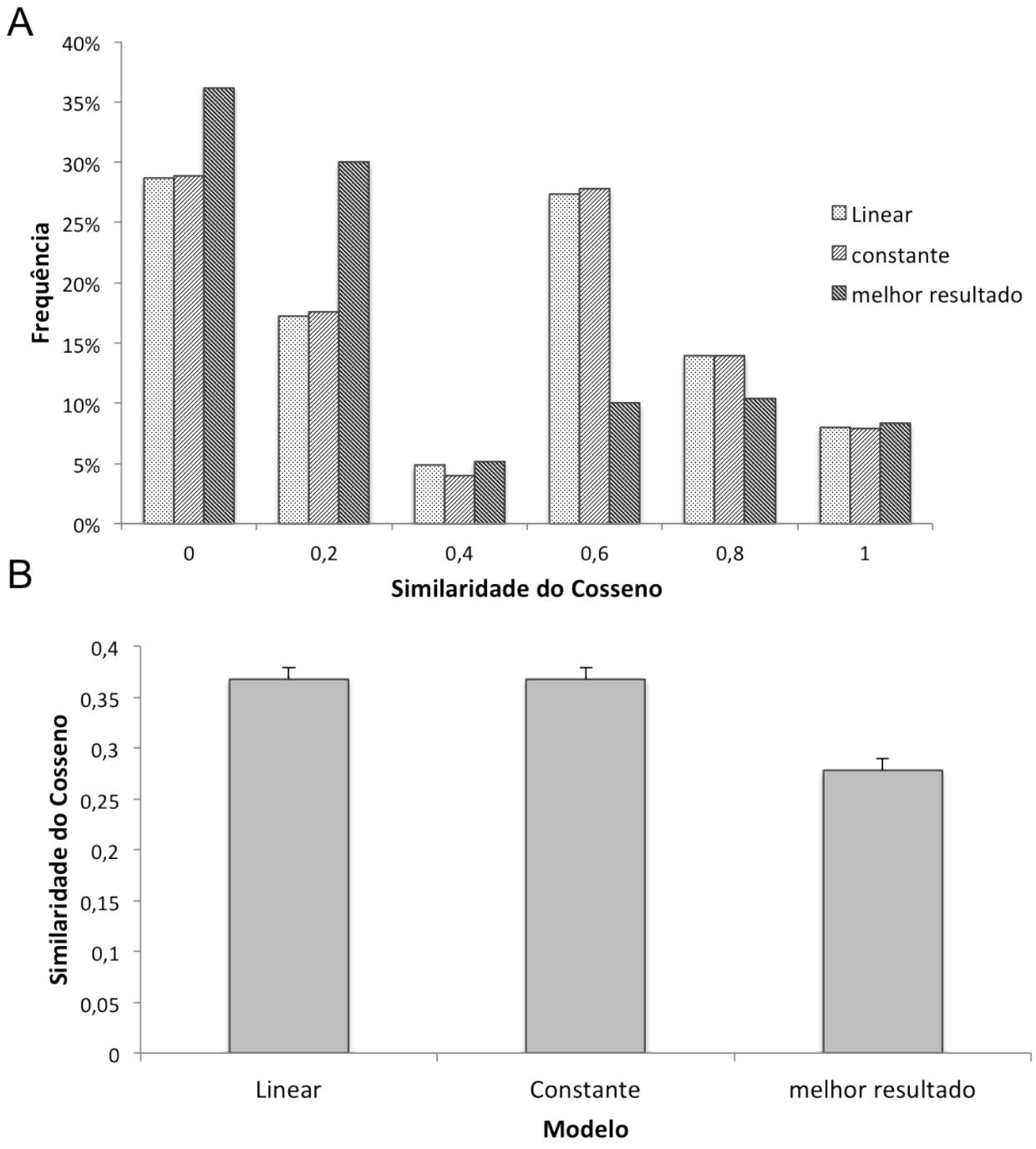

Figura 4.22: Gráficos mostrando similaridade do cosseno entre as anotações das *queries* e dos *subjects* selecionados por três métodos distintos a partir do modelo de BLASTp. A figura 'A' mostra a distribuição da similaridade do cosseno quando os *subjects* são selecionados pelo método de ponderação linear (linear), sem ponderação (constante) e seleção do *subject* com maior confiança (melhor resultado). A figura 'B' mostra a similaridade do cosseno médio entre as *queries* e os *subjcts* selecionados da forma já descrita. As barras de erro, no gráfico B, representam erro padrão.

o modelo de PSI-BLAST foi bem superior àquela usando o modelo de BLASTp, mas um fato importante que não é mostrado nestes gráficos é a quantidade de proteínas que receberam alguma sugestão de anotação, quer seja esta sugestão certa ou errada. O PSI-BLAST conseguiu sugerir anotação para as 900 proteínas enquanto o BLASTp sugeriu anotação para apenas 720. Isso mostra que algumas proteínas de função hipotética não podem ser anotadas com BLASTp, mas o podem com PSI-BLAST.

4.3 Annothetic

Com o conhecimento gerado nas seções anteriores, nós iremos agora construir uma ferramenta de análise de sequências chamada "Annothetic". Este nome vem da junção das palavras inglesas "*Annotate*" e "*hypothetic*". Essa ferramenta foi feita, primariamente, para sugerir anotação para proteínas hipotéticas. Essa pode ser usada para outros propósitos também, como encontrar o grupo de proteínas relacionadas a uma *query*, quer tenha esta função hipotética ou não.

A figura 4.23 mostra um *print-screen* da tela inicial do Annothetic. Na barra negra no alto da tela encontram-se três *links* sendo que o primeiro (*Home*) referencia a página atual. O segundo (*My Jobs*) faz referência a uma página que pode ser usada para recuperar resultado de execuções anteriores através do PID (*process id*). O terceiro, *link About Annothetic*, abre uma página com informações sobre o Annothetic, e como usa-lo. No meio da página vemos o logo do Annothetic bem como um retângulo cinza com um *link* (*example*), uma caixa de texto e dois botões. Apertar o link *example* preenche a caixa de texto com a sequência FASTA que pode ser vista na figura. É quase desnecessário dizer que o Annothetic aceita qualquer sequência em formato FASTA que for inserida na caixa de texto. O botão *clear* apaga a sequência FASTA, exceto se for a sequência "*example*". E o botão *submit*, não supreendentemente, submete a sequência para nossos procedimentos de PSI-BLAST e mineração bioinformática.

Um detalhe que pode ser visto no FASTA dentro da caixa de texto é que no final da primeira linha (cabeçalho da sequência) está escrito "taxid=64091". Isso informa para a ferramenta qual o táxon id da *query* (neste caso 64091). Essa informação não vem no FASTA exemplo e nem é necessária, porém a ausência dela pode comprometer o cálculo de LCA. Não obstante, nenhum outro cálculo, como a confiança, é comprometido ou sequer alterado.

Ao pressionar o botão "*submit*", o retângulo cinza com área de texto é substituído pelo retângulo da figura 4.24A. Como se pode ver, dentro dessa área cinza está escrito o PID, que é sempre um número inteiro longo. Caso a página do Annothetic seja fechada, o processo continua executando no servidor e os resultados podem ser recuperados posteriormente com este PID. Para isso, basta entrar na página "*My Jobs*". Complementarmente existe nesta área cinza um *link* (*Check HERE your results*). Esse *link* abre a página de espera mostrada na figura 4.24B. Essa página é atualizada automaticamente para a pagina de resultados (4.25) quando esses ficam prontos. Mais uma vez nessa página (4.24B), o PID pode ser visto.

A figura 4.25 mostra os resultados encontrados para a *query* submetida, nesse caso "Q9HMT0" de *Halobacterium salinarium*, uma proteína hipotética. Como se pode ver na figura, o resultado vem na forma de uma tabela. Na primeira coluna da tabela encontra-se o número da iteração, do PSI-BLAST, em que o *subject* apareceu pela primeira vez. Na segunda coluna, aparece o id do *subject*. Na terceira coluna temos a identidade, calculada pelo BLAST, entre o *subject* e a *query*. Na quarta coluna, temos o tamanho do alinhamento, também oriundo do BLAST. Na quinta coluna, temos a distância filogenética entre a *query* e o *subject*, calculado pelo MUSCLE. Na sexta coluna, temos a auto-pontuação da *query* naquela iteração de PSI-BLAST. Na sétima coluna, temos a confiança calculada pelo nosso *ensemble* de modelos de aprendizado de máquina. Na oitava coluna, temos o "*E-Value*" do *subject* calculado pelo BLAST. Na nona e décima colunas, temos o LCA, sendo que na décima encontra-se o nível da árvore que reúne a *query* e o *subject*.

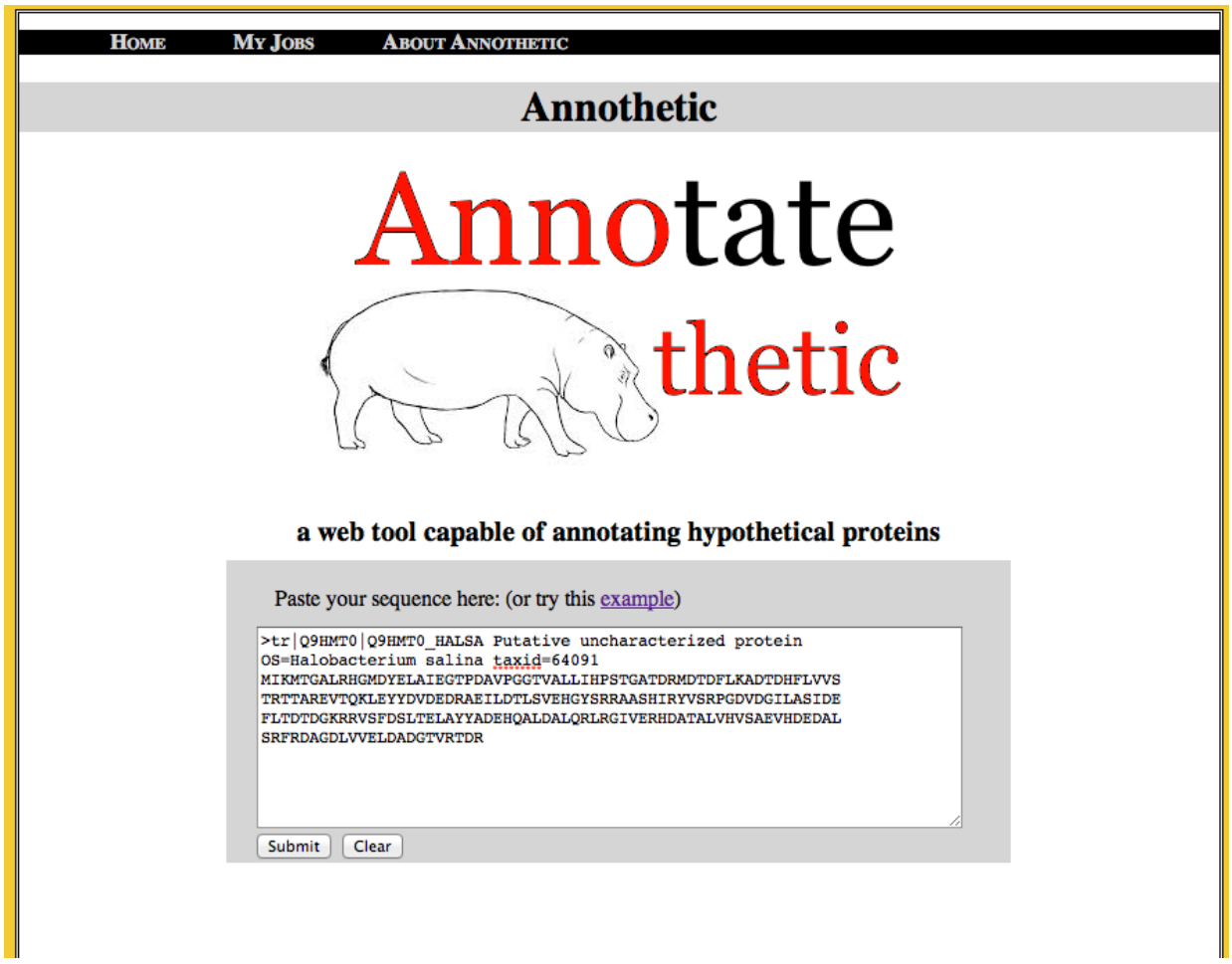

Figura 4.23: Figura mostrando a página inicial do Annothetic com a sequência de exemplo da ferramenta. A figura mostra uma caixa de texto com a sequência FASTA que aparece se o *link "try this example"* for pressionando. Observa-se no cabeçalho da sequência FASTA a presença do marcador 'taxid=64091'. Este Marcador não vem na sequência e é opcional. Ele informa explicitamente o táxon id da *query*. Em baixo da caixa de texto pode ser visto os botões *clear* e *submit*. O primeiro apaga o que estiver escrito na área de texto e o segundo submete a sequência FASTA para a processamento.

Esses níveis variam de 0 a 18 onde ”0” é *celular organism*, ”1” é super-reino, ”2” reino, ”3” filo, ”16” espécie, ”17” subespécie e ”18” genoma. A coluna nove mostra o táxon que reúne *query* e *subject*. A coluna 11 contém o nome do organismo de origem do *subject*. A coluna 13 contém a descrição UniProt do *subject* e a coluna 12 contém a ”relevância” desta descrição. Na verdade, esta coluna mostra o cosseno do ângulo entre a anotação presente e o consenso, conforme foi explicado na seção de mineração de anotação. Observe que as anotações hipotéticas não possuem cosseno e estão como ”NaN”.

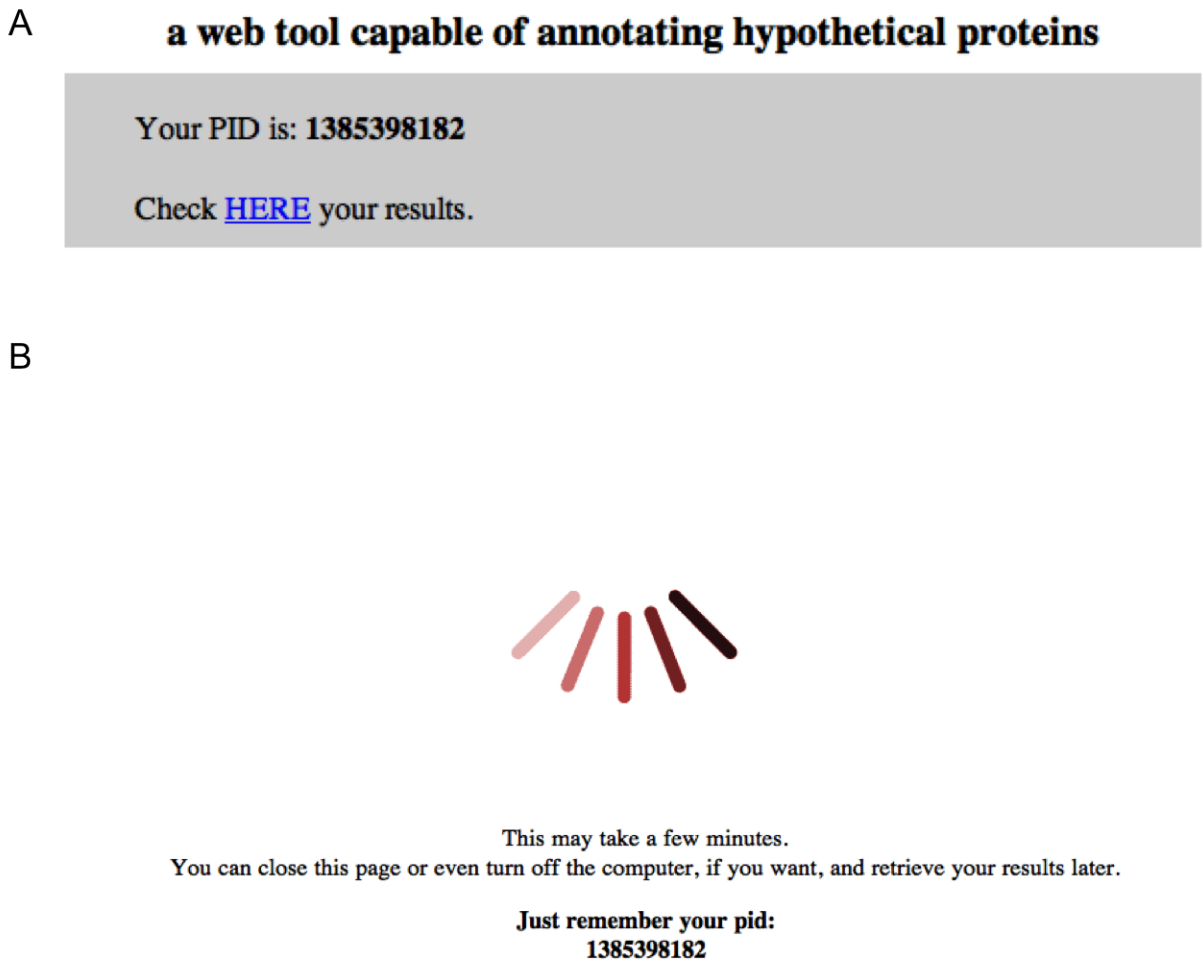

Figura 4.24: Figura mostrando as páginas que o usuário irá encontrar se submeter uma sequência no Annothetic. A figura A mostra a página que aparece logo que o botão *submit* (figura anterior) é pressionado. Nota-se nesta figura o PID que pode ser usado para recuperar os resultados posteriormente e um *link Check HERE your results*. A figura B mostra a página de espera que o usuário irá encontrar se pressionar o *link Check HERE your results* da figura A. Observa-se novamente o PID e uma 'barra' de espera.

Anteriormente, foi dito que não explicitar o táxon id da *query* no fasta pode comprometer os resultados de LCA. São os valores das colunas 9 e 10 que podem ficar comprometidos neste caso. Porém, é possível saber se este resultado foi comprometido, ou não. Se ele estiver correto, o LCA (coluna 10) da *query* deverá ser 18 ou no mínimo 16, uma vez que 'a *query* é da mesma espécie que ela mesma'. Caso contrário o LCA estará errado. Portanto, basta procurar o id da

query na tabela e checar se seu LCA é maior ou igual a 16. Não obstante, o LCA não é usado para nenhum outro cálculo e um erro de LCA não compromete nada além do próprio LCA.

Um outro fato que se pode visualizar na figura 4.25 é que algumas linhas estão em negrito. Essas são as linhas cuja anotação contém, pelo menos, uma palavra não *stopWord* e, portanto, são consideradas não hipotéticas. Um outro dado que pode ser visto na figura e que vale ressaltar é que esta tabela é reordenável. Ao se apertar o botão do *mouse* em cima das palavras do cabeçalho esta é reordenada pelo campo escolhido.

Como se pode ver na figura 4.23, submetemos uma proteína com anotação "*Putative uncharacterize protein*" e esta recebeu a sugestão de anotação de "*RecA-superfamily ATPase implicated in signal transduction-like protein*" (4.25). Esta foi a anotação retornada tanto pelo *best-hit* quanto pelo consenso ponderado das anotações. Podemos ver que o melhor resultado teve uma confiança de 0.79 e que este é corroborado por muitas outras anotações semelhantes com confiança variando de 0.69 a 0.2. Embora não se possa ver na figura, é importante ressaltar, que nenhuma anotação foi encontrada para esta proteína na iteração "1" do PSI-BLAST, que corresponde ao BLASTp, mostrando que esta proteína só pode receber sugestão de anotação com PSI-BLAST. E esse foi um dos motivo pelo qual escolhemos esta sequência para ser a sequência exemplo do Annothetic.

HOME MY JOBS ABOUT ANNOTHETIC

Annothetic

The lines written in green are proteins with annotation (non hypothetical).
Click on the table header to reordering.

Check the PhyloTaxonomicTree

Iteration	UniProt	Identity	Alignment	Filogenetic Distance	SelfScore	Confidence	E-value	LCA TxID	LCA	Organism	Description relevance	Description
3	D5E7T7	19.39	196	2.64557	0.7562	0.7886	1.0E-55	0	3	Methanohalophilus mahii DSM 5219	0.74	D5E7T7_METMS RecA-superfamily ATPase implicated in signal transduction
3	F2KQ27	21.31	183	2.71674	0.7562	0.6862	5.0E-39	0	3	Archaeoglobus veneficus SNP6	0.74	F2KQ27_ARCVS RecA-superfamily ATPase implicated in signal transduction
3	D2EFY2	17.61	176	3.34826	0.7562	0.4445	3.0E-11	0	3	Candidatus Parvarchaeum acidiphilum ARMAN-4	0.74	D2EFY2_9EURY RecA-superfamily ATPase-like protein implicated in signal transduction
4	D6GW22	16.48	182	3.32108	0.6244	0.3650	6.0E-24	0	3	Candidatus Parvarchaeum acidophilus ARMAN-5	0.74	D6GW22_9EURY RecA-superfamily ATPase implicated in signal transduction-like protein
3	B5I9Q8	21.23	179	3.85011	0.7562	0.2323	8.0E-13	0	3	Aciduliprofundum boonei T469	0.74	B5I9Q8_ACIB4 RecA-superfamily ATPase implicated in signal transduction-like protein
5	D2RH39	14.66	191	4.41084	0.5547	0.2203	5.0E-14	0	3	Archaeoglobus profundus DSM 5631	0.74	D2RH39_ARCPA RecA-superfamily ATPase implicated in signal transduction-like protein
5	C5A4F6	16.5	200	4.04128	0.5547	0.1804	6.0E-14	0	3	Thermococcus gammatolerans EJ3	0.74	C5A4F6_THEGJ RecA-superfamily ATPase implicated in signal transduction, putative
5	D5U170	15.96	188	4.1991	0.5547	0.1800	4.0E-14	0	0	Thermosphaera aggregans DSM 11486	0.74	D5U170_THEAM RecA-superfamily ATPase implicated in signal transduction
5	F8AH76	15.0	200	4.21113	0.5547	0.1504	5.0E-11	0	3	Pyrococcus yayanosii CH1	0.74	F8AH76_PYRYC RecA-superfamily ATPase implicated in signal transduction, putative
5	F4XW39	19.57	184	5.41803	0.5547	0.1031	5.0E-11	0	0	Moorea producens 3L	0.74	F4XW39_9CYAN RecA-superfamily ATPase implicated in signal transduction

Figura 4.25: Figura mostrando a página com os resultados que o Annothetic encontrou para a sequência exemplo. Observa-se na figura uma tabela com o número da iteração do PSI-BLAST na coluna 1, os ids dos *subjects* na coluna 2, identidade na coluna 3, tamanho do alinhamento na coluna 4, distância filogenética na coluna 5, auto-pontuação na coluna 6, confiança calculada pelo nosso modelo na coluna 7, *e-value* na coluna 8, LCA nas colunas 9 e 10, organismo do *subject* na coluna 11, "índice" de concordância da anotação com o consenso na coluna 12 e a anotação na coluna 13. Observa-se ainda, acima da tabela, o *link Check the PhyloTaxonomicTree* que abre a página com a árvore filogenética. Por fim pode-se observar que a tabela é reordenável.

Embora esta proteína seja considerada hipotética, a base de dados EggNOG sugere que esta proteína seja uma *"RecA-superfamily ATPase implicated in signal transduction-like protein"* conforme nosso modelo sugeriu.

4.4 Um estudo de caso com Amino-acil-tRNA ligases

Na seção anterior um breve estudo de caso foi feito com uma proteína hipotética de *Halobacterium sp.* e mostramos como nossa ferramenta é capaz de sugerir anotação para ela. Nesta seção, por outro lado, faremos uma breve demonstração de como nossa ferramenta se comporta com um grupo de proteínas bem estudado. Nós selecionamos para este teste 13 amino-acil-tRNA ligases de *Halobacterium salinarium* e as submetemos a Annothetic. Em geral, neste caso, o Annothetic deveria retornar anotação com BLASTp, mas na metodologia utilizada nós o obrigamos a fazer PSI-BLAST para observar como o modelo se comporta.

A tabela 4.3 mostra os resultados obtidos. Na primeira coluna, observam-se os ids das *queries*, na segunda coluna os ids dos *subjects*, na terceira coluna a confiança, na quarta o cosseno da anotação com o consenso ponderado e na quinta coluna a anotação. Como se pode ver nesta tabela, todas as 13 tRNAs ligases foram, não surpreendentemente, corretamente anotadas. Esse resultado foi obtido tanto pelo *best-hit* quanto pelo consenso. Além disso os valores de cosseno (coluna 4) elevados mostram que houve um consenso alto entre as descrições dos *subjects* recrutados.

Além dos dados observados na tabela, também medimos a precisão do Annothetic sobre *EC-numbers*. Neste caso, *subjects* com *EC-numbers* iguais ao da *query* foram considerados positivos. *Subjects* com *EC-numbers* diferentes ao da *query* foram considerados negativos e *subjects* sem *EC-numbers* foram descartados. Os resultados obtiveram uma precisão de mais de 96.6%, ou seja, 96.6% das proteínas recrutadas pelo nosso modelo, independente da confiança, tinham o mesmo *EC-numbers* da *query*. Esse experimento também foi repetido com os KOs das 13 proteínas e a precisão também atingiu a mesma taxa.

Estes dados podem parecer um pouco estranhos a princípio pois uma precisão alta foi encontrada, independente da confiança do nosso modelo. No entanto, estas tRNA ligases são um grupo de proteínas bem estudado e bem coeso, onde o PSI-BLAST não inclui falsos-positivos. Nosso modelo, por outro, lado foi treinado em um amplo espectro de *clusteres* de proteínas. *Clusteres*, estes, que nem sempre são tão "bem comportados" como as tRNA ligases e o nosso modelo aprendeu a dar confiança baixa para aquelas proteínas "suspeitas". De qualquer forma, vale ressaltar o óbvio: submeter proteínas "bem comportadas" para o Annothetic não gera nenhum problema para o algoritmo, muito pelo contrário, como se pode ver na tabela 4.3, todas as tRNAs foram corretamente classificadas com confiança bem altas.

Tabela 4.3: Tabela mostrando a sugestão de anotação para 13 aminoacil-tRNA-ligases de *Halobacterium sp.*. Na primeira coluna pode-se ver o id da *query*, na segunda coluna o *subject*, na terceira coluna a confiança, na quarta coluna o cosseno da anotação em relação ao consenso e na quina coluna a anotação sugerida.

Query	*Subject*	Confiança	Cosseno	Anotação
Q9HN24	D3S7L8	0.9966	0.99	Alanine–tRNA ligase
O07683	Q6HJE8	0.9977	0.88	Aspartate–tRNA ligase (Aspartyl-tRNA synthetase)
Q9HHN2	B0R9X1	0.9861	0.98	Arginine–tRNA ligase
Q9HN97	P26499	0.9911	1.00	Isoleucine–tRNA ligase
Q9HNP5	Q0W928	0.9983	0.99	Histidine–tRNA ligase
Q9HSA4	Q0AAG6	0.9707	0.99	Methionine–tRNA ligase
Q9HN62	Q46BQ5	0.9984	0.79	Tyrosine–tRNA ligase
Q9HNJ8	D4GTF0	0.9954	0.99	Serine–tRNA ligase
Q9HN83	A0B637	0.9908	0.89	Tryptophanyl-tRNA synthetase
Q9HN66	D1YYF9	0.9970	0.92	Tryptophanyl-tRNA synthetase
Q9HP27	Q9WZJ9	0.8883	1.00	Threonine–tRNA ligase
Q9HN72	F8DWB3	0.9925	0.98	Leucine–tRNA ligase
Q9HMG9	B0R841	0.9618	0.99	Valine–tRNA ligase

4.5 Experimento Com Quatro Microrganismos

Nesta seção aplicaremos as técnicas desenvolvidas anteriormente em um estudo de caso com quatro microrganismos, a saber: *Escherichia coli*, *Bacilus subtilis*, *Micobacterium tuberculosis* e a arquea *Halobacterium sp.*

4.5.1 Escherichia coli

A figura 4.26 mostra uma descrição resumida dos resultados de nossos procedimentos em proteínas hipotéticas de *Escherichia coli* . A figura 4.26A mostra quantas proteínas receberam alguma sugestão de anotação com BLASTp, com PSI-BLAST e quantas não receberam nenhuma sugestão. Pode-se ver nessa figura, de um total de 451 proteínas consideradas hipotéticas em *E. coli*, 409 receberam alguma sugestão de anotação, sendo que destas apenas seis receberam a sugestão apenas com o modelo de PSI-BLAST. Estas sugestões variam desde uma função da proteína até sua localização. É claro que estamos mais interessados em função, porém, qualquer informação sobre sua localização e/ou natureza físico-química também é importante.

A figura 4.26A não nos diz, no entanto, o quão informativas e detalhadas estas sugestões de anotação são. Em outras palavras, as 409 sugestões poderiam ser todas formadas por descrições muito pobres, como uma sigla do tipo "Ydda" que não foi considerada *stopWord*, ou poderiam ainda ser do tipo *membrane protein* que é "melhor do que nada" mas está longe de ser uma anotação completa. A figura 4.26B mostra uma estimativa deste "detalhamento" das anotações.

Conforme foi dito nos materiais e métodos (seção 3.9.2), esta figura foi construída iterativamente da seguinte forma: primeiro excluímos um grupo de palavras e contamos quantas anotações ficaram vazias, com isso uma faixa do gráfico de pizza (4.26B) foi criada, então excluímos outro grupo de palavras e contamos quantas anotações a mais ficaram vazias e assim por diante.

O primeiro destes grupos de palavras que nós excluímos foram as palavras com menos de cinco letras e siglas que misturam letra e número porque isso da uma estimativa de quantas anotações foram feitas apenas com siglas "crípticas". Ao contar quantas anotações ficaram vazias após esta exclusão, a fatia "siglas" (4.26B) foi criada. Como se pode ver no gráfico, 96 proteínas foram anotadas apenas com estas siglas.

Em seguida, excluímos algumas palavras que descrevem localização como *"membrane"*, *"nuclear"* e *"cytoplasmic"* e contamos quantas novas anotações ficaram vazias e assim definimos a fatia "localização". Podemos ver que para 66 proteínas hipotéticas a única sugestão de anotação foi sua localização, como por exemplo se a proteína é de membrana, citossólica, nuclear entre outros.

O próximo passo foi excluir as palavras, *"lipoprotein"*, *"metaloprotein"* e *"selenoprotein"* e a contagem de novas anotações que ficaram vazias gerou a fatia "*proteins*". O gráfico mostra que 21 proteínas foram anotadas com estas palavras. Estas anotações assim como as anteriores (localização) não nos dizem a função da proteína, mas nos dizem algo sobre suas características físico-químicas.

Por fim, contamos quantas das anotações remanescentes continham palavras terminadas

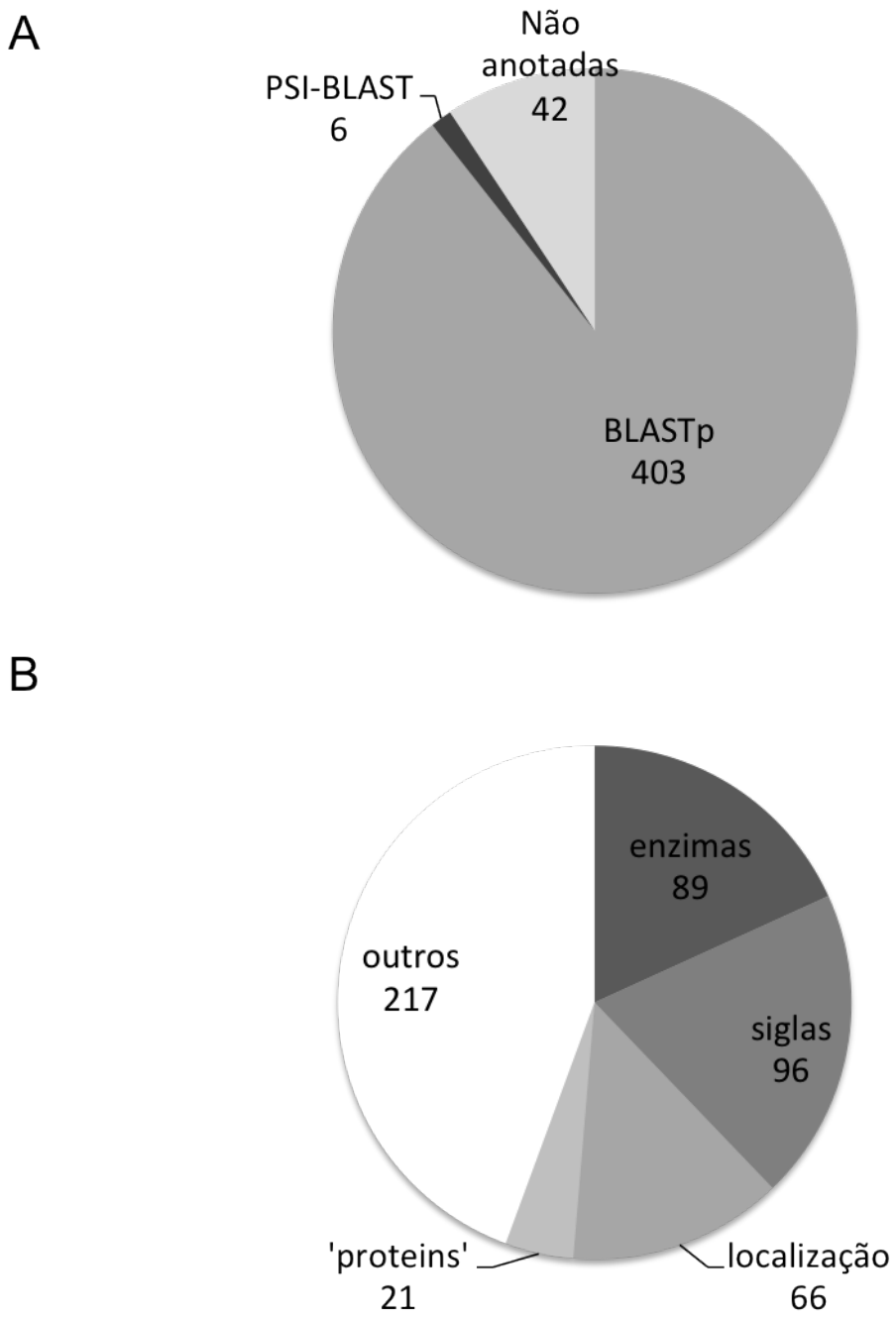

Figura 4.26: Gráficos mostrando os resultados de anotação de proteínas hipotéticas de *Escherichia coli*. O gráfico A mostra a quantidade de proteínas que receberam alguma sugestão de anotação com BLASTp e PSI-BLAST bem como o número daquelas que não receberam nenhuma sugestão. O gráfico B mostra a distribuição das anotações por tipo. Sendo que 'enzimas' são anotações relacionadas a enzima; 'siglas' são proteínas anotadas com siglas pouco informativas; 'localização' são proteínas anotadas apenas como *membrane, citoplasmic, nuclear, citosolic* ou extracelular; 'proteins' são anotadas apenas como metalo-proteínas, seleno-proteínas ou lipo-proteínas; 'outros' são outras anotações que não se encacham em nenhuma das anteriores.

em ”ase” e esta contagem gerou a fatia ”enzimas” e o complemento desta contagem gerou a fatia ”outros”. Observe que esta última etapa foi diferente das anteriores pois fizemos uma contagem direta sem excluir palavras. Podemos ver, então, que 89 sugestões de anotação são enzimas, proteínas semelhantes a enzimas ou proteínas relacionadas a enzimas. Em síntese, o que estes resultados mostram é que nosso modelo é capaz de fazer sugestões de anotações variadas e com níveis variados de detalhamento.

Antes de passarmos para o próximo resultado, algumas considerações precisam ser feitas. Na fatia de ”siglas” contamos anotações que tinham apenas siglas com menos de cinco letras e siglas misturando letras e números. De fato, a maioria das ”siglas-crípticas” ou possuem menos de cinco letras ou misturam letra e número, porém, algumas siglas com ”RNA”, ”ATP”, ”NADH” entre outras também são indesejadamente contadas. No entanto, estas siglas (como ”RNA”) raramente aparecem sozinhas na anotação de uma proteína e, por isso, elas acabam não sendo contadas na fatia de ”siglas”, porque ao se excluir estas siglas ainda restam outras palavras. Por exemplo, na anotação ”*16S ribossomal RNA methyltransferase*” as palavras ”16S” e ”RNA” são excluídas mas ainda sobra *”ribossomal”* e *”methyltransferase”* e se seguir o raciocínio apresentado anteriormente esta anotação irá acabar sendo considerada, corretamente, como enzima.

Como se pode ver na figura 4.26A apenas seis proteínas de *E. coli* receberam anotação, com confiança acima de 0.5, com PSI-BLAST. A tabela 4.4 mostra as sugestões de anotação que essas proteínas receberam. Essa tabela mostra os ids das *queries* e dos *subjects*, a confiança do *subjects*, a anotação do *subjects* e, adicionalmente anotações sugeridas para essa proteína em outros bancos de dados. É importante ressaltar que essas seis proteínas eram consideradas hipotéticas pelo UniProt em janeiro de 2013 mas outros bancos de dados, como Pfam, GO e eggNOG, podem ter alguma sugestão de anotação. Além do mais, a coluna ”anotação em outros bancos de dados” foi feita com dados de outubro de 2013. Portanto, algumas destas proteínas poderiam ter anotação mesmo no UniProt.

A tabela 4.4 mostra que três das seis proteínas mostradas receberam sugestão de anotação, em outros bancos de dados, idêntica à que nós sugerimos, sendo duas delas sugeridas pelo Pfam e corroborado pelo InterPro. É um bom resultado que nosso modelo esteja sugerindo anotação em acordo com o que outros bancos, já bem sedimentados, estão sugerindo. Também é bom ver que nosso modelo sugere anotação para proteínas que estes bancos não sugerem.

Tabela 4.4: Tabela mostrando as seis sugestões de anotação de *Escherichia coli* encontradas apenas com PSI-BLAST. A coluna 1 mostra o id da *query* a coluna 2 o id do *subject* a coluna 3 a confiança a coluna 4 a anotação do *subject* e a coluna 5 a sugestão de anotação feita por outros bancos de dados ou mesmo por um UniProt mais recente (se existir).

Query	*Subject*	Confiança	Anotação sugerida	Anotação em outros bancos de dados
B1XAU3	Q8Z7Y7	0,8766	Putative bacteriophage protein	(eggNOG) Putative bacteriophage protein
B1XBG9	G1Y8E6	0,5410	Valyl-tRNA synthetase domain protein	
B1XGI2	K4YJP8	0,5390	Putative transmembrane anchor protein	
B1XGI2	F9EW00	0,5333	Ankyrin repeat protein	
B1XCT2	D3RPL6	0,8807	CRISPR-associated protein Cse3 family	(InterPro) CRISPR-associated protein Cse3
B1XDH2	F4MB34	0,9811	CRISPR-associated protein Cse2	(InterPro) CRISPR-associated protein Cse2; (Pfam) CRISPR-associated protein Cse2

4.5.2 Bacilus subtilis

A figura 4.27 resume os resultados de anotação de proteínas hipotéticas de *Bacilus subtilis*. Como se pode ver na figura 4.27A, o *B. subtilis* tem 429 proteínas hipotéticas e nosso modelo foi capaz de sugerir anotação para 176 dessas proteínas, sendo que 116 proteínas receberam alguma sugestão de anotação com o modelo de BLASTp e 60 com o modelo de PSI-BLAST. Ao se comparar este resultado com o da *E. coli* observa-se que uma quantidade maior de proteínas receberam anotação apenas com PSI-BLAST, fazendo deste mais importante.

A figura 4.27B mostra uma estimativa dos níveis de detalhamento destas anotações. Como se pode ver, 49 anotações encontradas por nosso modelo são ou estão relacionadas a enzimas e 23 dão apenas informação sobre a localização das proteínas. Mais uma vez este resultado mostra que nosso modelo foi capaz de sugerir anotação detalhada para muitas proteínas.

Na tabela 4.5 são mostradas as sugestões de anotação para 10 proteínas selecionadas. Nesta tabela, podemos ver os ids das *query* e dos *subjects*, a confiança do *subject* e sua anotação. Também se pode ver sugestões de anotação feita por outros bancos de dados. Como se observa nestes resultados, sete das 10 proteínas selecionadas têm anotação sugerida por outros bancos de dados. A tabela parece mostrar oito confirmações em outros bancos, mas iremos mostrar que, na verdade, são apenas sete. Destas, seis confirmam o que nós sugerimos, incluindo sugestões de enzimas. Mas existe uma "anotação" na quinta linha da tabela feita pelo Pfam que desmente totalmente o que nosso modelo sugeriu. O que nós sugerimos foi "*Aminodeoxichorismate lyase*", mas o Pfam diz "família de proteínas previamente anotada incorretamente como '*Aminodeoxichorismate lyase*"'. Portanto, neste caso, nosso modelo errou, e isso é natural, pois não estamos desenvolvendo um oráculo que não erra nunca. Mas é bastante interessante este erro, pois, esta proteína já teve, de fato, a anotação que nós sugerimos e isso só é possível porque ela deve se parecer com uma "*Aminodeoxichorismate lyase*", Ou seja, nosso modelo não errou 100%.

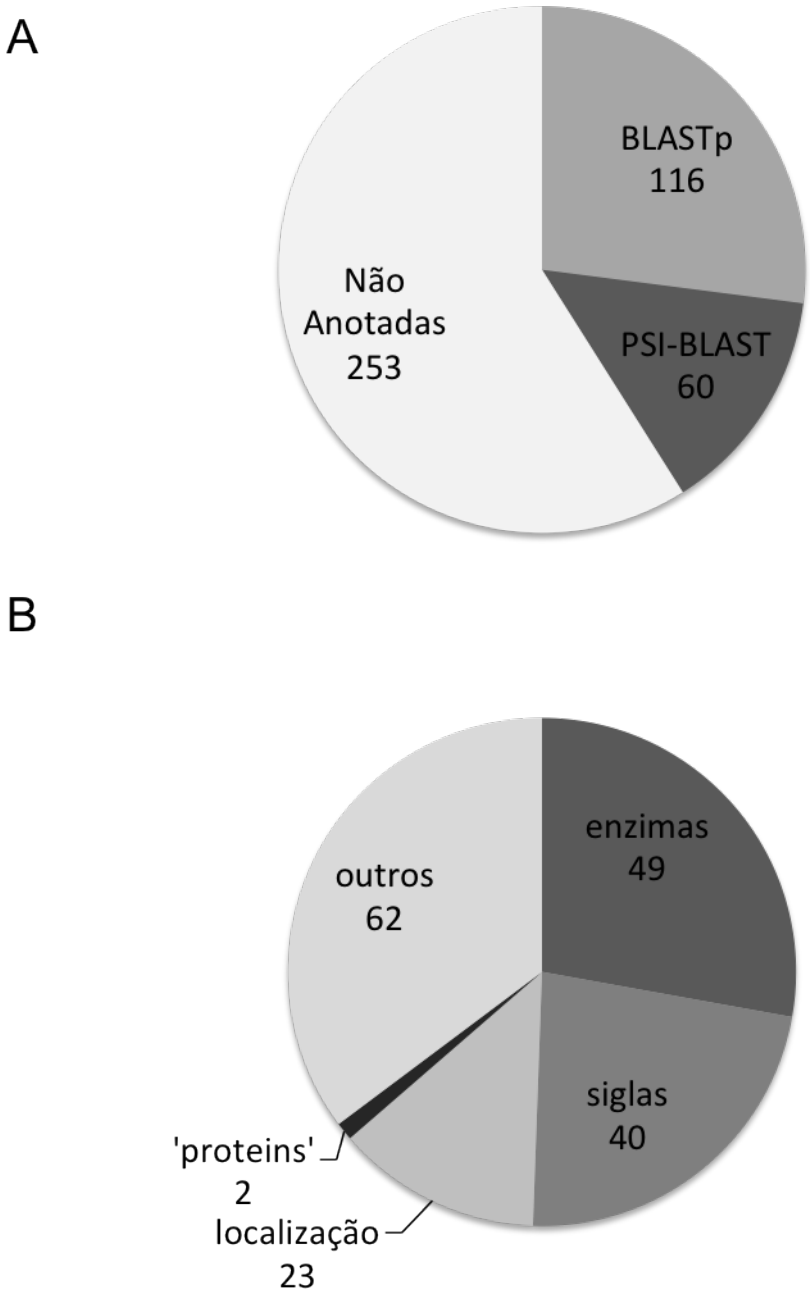

Figura 4.27: Gráficos mostrando os resultados de anotação de proteínas hipotéticas de *Bacilus subtilis*. O gráfico A mostra a quantidade de proteínas que receberam sugestão de anotação com BLASTp e PSI-BLAST bem como o número daquelas que não receberam nenhuma sugestão. O gráfico B mostra a distribuição das anotações por tipo. Sendo que 'enzimas' são anotações relacionadas a enzima; 'siglas' são proteínas anotadas com siglas pouco informativas; 'localização' são proteínas anotadas apenas como *membrane, citoplasmic, nuclear, citosolic* ou extracelular; 'proteins' são anotadas apenas como metalo-proteínas, seleno-proteínas ou lipo-proteínas; 'outros' são outras anotações que não se encacham em nenhuma das anteriores.

Tabela 4.5: Tabela mostrando as 10 sugestões de anotação sorteadas de *Bacilus subtilis*. A coluna 1 mostra o id da *query* a coluna 2 o id do *subject* a coluna 3 a confiança a coluna 4 a anotação do *subject* e a coluna 5 a sugestão de anotação feita por outros bancos de dados ou mesmo por um UniProt mais recente (se existir).

Query	*Subject*	Confiança	Anotação sugerida	Anotação em outros bancos de dados
O06727	K0FID7	0,6674	S1 RNA-binding domain-containing protein	
O35012	G4P5H8	0,9185	Peptidase propeptide and ypeb domain protein	(Pfam) Peptidase propeptide and YPEB domain
O34495	F6CH20	0,6967	Polysaccharide deacetylase	
Q45478	C2PJ58	0,6569	Transcriptional regulator TraR/DksA	(GO) zinc ion binding; (InterPro) Transcription regulator DksA-related; (Pfam) Prokaryotic dksA/traR C4-type zinc finger;
O32023	G8N281	0,7826	Aminodeoxychorismate lyase	(Pfam) familia previamente anotada incorretamente como aminodeoxychorismate lyase
P37481	D8GVC2	0,7529	Ferrichrome transport system permease protein fhuB	
P45948	Q65KQ3	0,9977	Bipartite response regulator C-terminal effector	(GO) regulation of transcription, DNA-dependent; DNA binding; sequence-specific DNA binding transcription factor activity; sigma factor activity
P37496	Q02ZX8	0,9977	Ketosteroid isomerase related protein	(eggNOG) Ketosteroid isomerase-like protein
O07909	Q4WVY3	0,5617	NlpC/P60-like cell-wall peptidase putative	(Pfam) SH3 domain
O32180	F7Z5T6	0,9977	Coat F domain protein	(Pfam) Coat F domain

4.5.3 Halobacterium sp.

A figura 4.28 resume os resultados de anotação para proteínas hipotéticas de *Halobacterium sp.*. Como se pode ver no gráfico 4.28A, essa bactéria contém 1394 proteínas sem qualquer anotação e nosso procedimento foi capaz de sugerir anotação para 670 delas. Dessas sugestões de anotação, 409 foram obtidas com PSI-BLAST. Este resultado mostra que, para alguns organismos, como as halobactérias, a maioria das proteínas não-caracterizadas não podem ser anotada, com confiança, apenas com BLASTp.

A figura 4.28B mostra a estimativa dos níveis de detalhamento das anotações que foram encontradas. Diferente dos microrganismos anteriores, mais da metade das sugestões são relacionadas a enzimas e apenas uma quantidade bem pequena está relacionada a "siglas suspeitas" ou localização apenas.

Na tabela 4.6 podemos ver 10 anotações sorteadas dentre as 409 proteínas que foram anotadas apenas com PSI-BLAST. Podemos ver que todas as 10 proteínas selecionadas tiveram sua sugestão de anotação confirmada pelas bases de dados GO, Pfam e InterPro. Mais uma vez, o fato de nossos resultados concordarem com o que outras bases de dados dizem aumentam a confiança do nosso modelo.

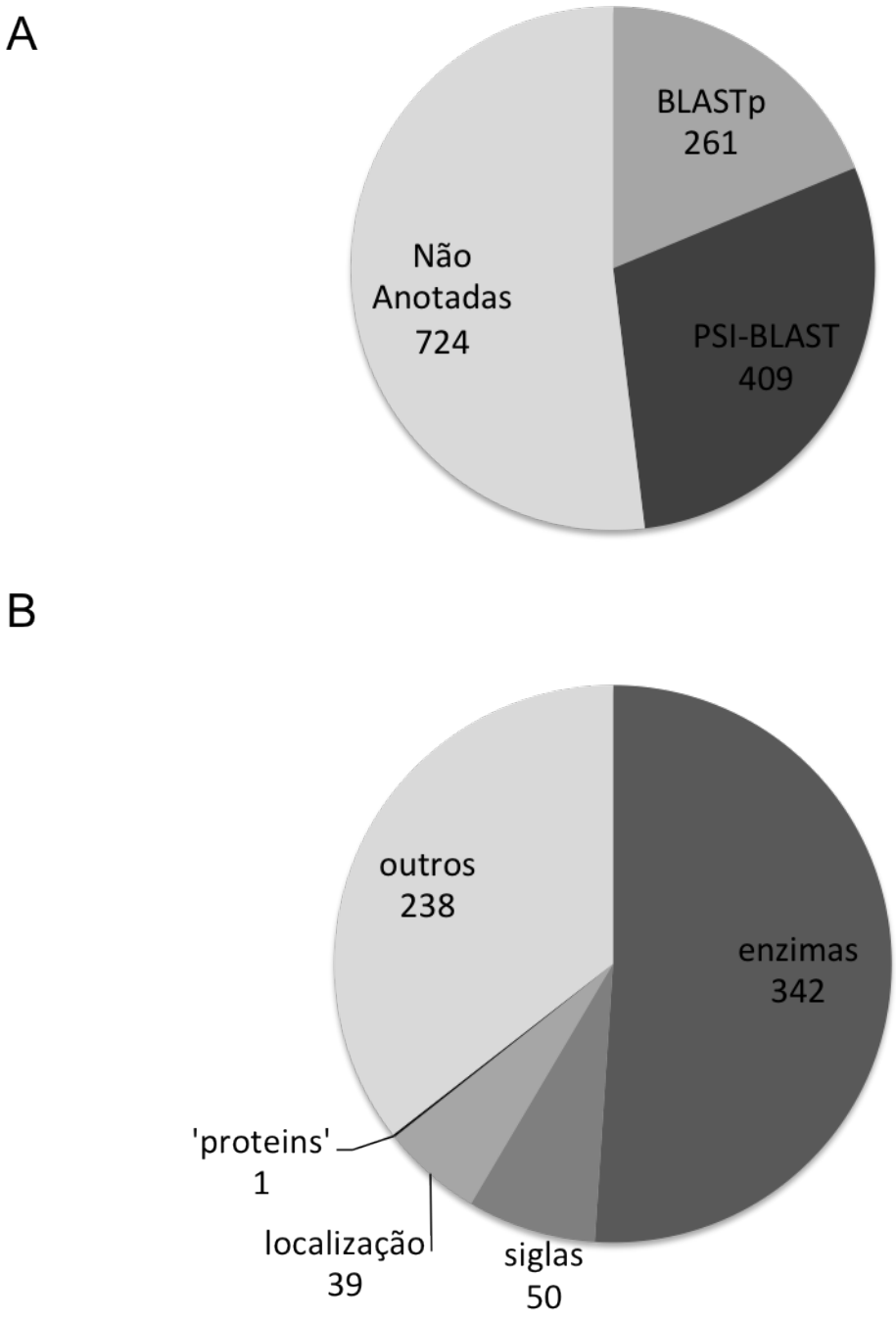

Figura 4.28: Gráficos mostrando os resultados de anotação de proteínas hipotéticas de *Halobacterium sp*. O gráfico A mostra a quantidade de proteínas que receberam sugestão de anotação com BLASTp e PSI-BLAST bem como o número daquelas que não receberam nenhuma sugestão. O gráfico B mostra a distribuição das anotações por tipo. Sendo que 'enzimas' são anotações relacionadas a enzima; 'siglas' são proteínas anotadas com siglas pouco informativas; 'localização' são proteínas anotadas apenas como *membrane, citoplasmic, nuclear, citosolic* ou extracelular; 'proteins' são anotadas apenas como metalo-proteínas, seleno-proteínas ou lipo-proteínas; 'outros' são outras anotações que não se encacham em nenhuma das anteriores.

Tabela 4.6: Tabela mostrando as 10 sugestoes de anotação sorteadas de *Halobacterium sp.*. A coluna 1 mostra o id da *query* a coluna 2 o id do *subject* a coluna 3 a confiança a coluna 4 a anotação do *subject* e a coluna 5 a sugestão de anotação feita por outros bancos de dados ou mesmo por um UniProt mais recente (se existir).

Query	*Subject*	Confiança	Anotação sugerida	Anotação em outros bancos de dados
O51966	Q64B41	0,7878	Transposase	(GO) transposition, DNA-mediated; DNA binding; transposase activity
Q9HSJ2	G4IIA2	0,9680	Flagellin domain protein (Precursor)	(InterPro) Flagellin, N-terminal
Q9HSD0	E4NTI4	0,9970	Transcriptional regulator ArsR family	(InterPro) Winged helix-turn-helix DNA-binding domai; (eggNOG) Transcriptional regulator containing HTH domain, ArsR family
Q9HS62	J2ZI16	0,8694	Transcriptional regulator containing an hth domain fused to a zn-ribbon	(eggNOG) Predicted transcriptional regulator containing an HTH domain fused to a Zn-ribbon
Q9HRW5	Q2FTY5	0,9872	Transcriptional regulator ArsR family	(InterPro) Winged helix-turn-helix DNA-binding domain; (eggNOG) Transcriptional regulator containing HTH domain, ArsR family
Q9HRB1	D8J4S5	0,8048	Proteasome Rpn11 subunit JAMM motif protein	(Pfam) JAB domain, prokaryotic; (eggNOG) Predicted metal-dependent protease of the PAD1/JAB1 superfamily; JAMM-like protein
Q9HR03	I7CIG5	0,9821	Alpha-1 4-glucan-protein synthase UDP-forming	(GO) cellulose biosynthetic process; intramolecular transferase activity
Q9HQF8	D3T164	0,5045	Pyridoxamine 5'-phosphate oxidase-related FMN-binding protein	(GO) FMN binding; oxidoreductase activity
Q9HQ79	D9PWS2	0,9016	Predicted exosome subunit	(InterPro) Ribosomal protein L5 domain; (eggNOG) Predicted exosome subunit
Q9HPE0	G4GCK4	0,9780	Carotene biosynthesis associated membrane protein	(InterPro) Carotene biosynthesis associated membrane protein, type-2; (eggNog) carotene biosynthesis associated membrane protein

4.5.4 Micobacterium tuberculosis

A figura 4.29 resume os resultados de anotação da *Micobacterium teberculosis*. A figura 4.29A mostra que a *M. tuberculosis* possui 1380 proteínas consideradas hipotéticas. Dentre estas, 759 receberam alguma sugestão de anotação com nossos procedimentos, sendo que 169 receberam sugestão com PSI-BLAST. Como se pode ver, mais uma vez aqui, um grande número de proteínas não puderam ser anotadas com modelo de BLASTp, mas sim com o modelo de PSI-BLAST.

A figura 4.29B mostra uma estimativa do grau de detalhamento das anotações encontradas. Assim como na halobactéria, a maior parte das anotações sugeridas está relacionada a enzimas e apenas um pequeno número corresponde a anotações pobres como siglas suspeitas, localização da proteína e características físico-químicas genéricas como lipoproteína e metaloproteína.

A tabela 4.7 mostra 10 proteínas selecionadas *Micobacterium tuberculosis*. Destas seis possuíam alguma sugestão de anotação em outros bancos de dados. Dessas seis sugestões, pelo menos quatro confirmam o que nosso procedimento sugeriu. Mas uma em especial é bem importante: a proteína "Q7D9K6". No UniProt de janeiro de 2013 esta proteína estava descrita como "*putative uncharacterized protein*" e nosso procedimento fez uma busca neste banco de dados (UniProt 01/2013) e sugeriu que ela seria uma "*ribonuclease VapC*". Já o UniProt de outubro de 2013 sugeria que ela seria uma ribonuclease e incluiu até "EC-number" na descrição como se pode ver na tabela 4.7.

Os resultados mostrados para estes quatro microrganismos confirmam o poder da ferramenta desenvolvida. Como se pode ver, nossa ferramenta foi capaz de sugerir anotação para mais da metade das proteínas hipotéticas destes quatro organismos, sendo que uma boa parte se deu apenas com o modelo de PSI-BLAST.

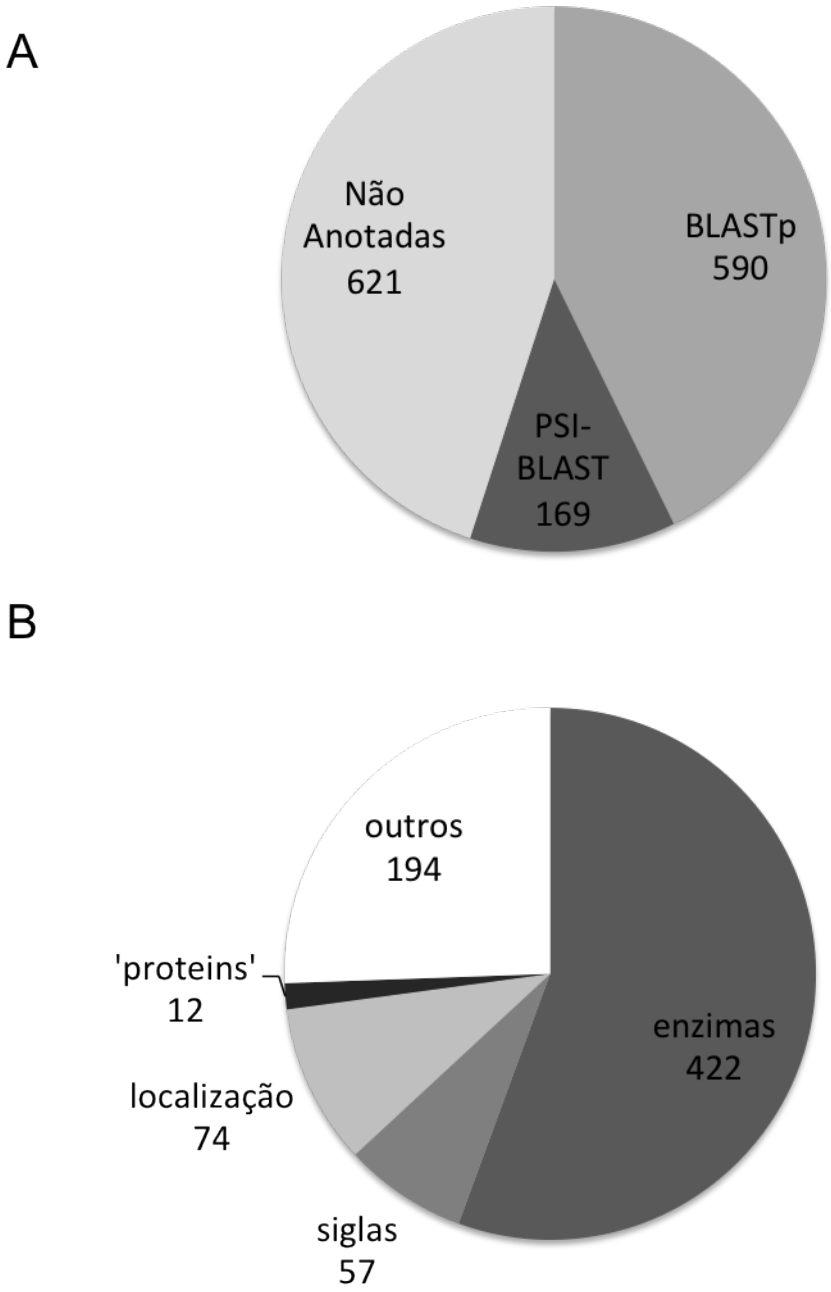

Figura 4.29: Gráficos mostrando os resultados de anotação de proteínas hipotéticas de *Micobacterim tuberculosis*. O gráfico A mostra a quantidade de proteínas que receberam sugestão de anotação com BLASTp e PSI-BLAST bem como o número daquelas que não receberam nenhuma sugestão. O gráfico B mostra a distribuição das anotações por tipo. Sendo que 'enzimas' são anotações relacionadas a enzima; 'siglas' são proteínas anotadas com siglas pouco informativas; 'localização' são proteínas anotadas apenas como *membrane, citoplasmic, nuclear, citosolic* ou extracelular; 'proteins' são anotadas apenas como metalo-proteínas, seleno-proteínas glico-proteínas ou lipo-proteínas; 'outros' são outras anotações que não se encacham em nenhuma das anteriores.

Tabela 4.7: Tabela mostrando as 10 sugestões de anotação sorteadas de *Micobacterium tuberculosis*. A coluna 1 mostra o id da *query* a coluna 2 o id do *subject* a coluna 3 a confiança a coluna 4 a anotação do *subject* e a coluna 5 a sugestão de anotação feita por outros bancos de dados ou mesmo por um UniProt mais recente (se existir).

Query	*Subject*	Confiança	Anotação sugerida	Anotação em outros bancos de dados
P71696	H8ET42	0,9982	Transmembrane protein	(GO) integral to membrane; plasma membrane
Q8VKQ3	Q7U2N3	0,9653	PROBABLE CONSERVED MCE ASSOCIATED MEMBRANE PROTEIN	
O07251	E3IWA1	0,9921	Methyltransferase type 11	(GO) methylation; methyltransferase activity
P64701	B8HFE8	0,5219	Proteinase inhibitor I25 cystatin	(GO) growth of symbiont in host cell
O53766	A1SDE4	0,6761	Putative signal-transduction protein with CBS domains	(GO) response to hypoxia
Q7D9K6	E2WEF9	0,9300	Probable ribonuclease VapC	Probable ribonuclease VapC EC=3.1.-.-
Q7D9H0	K0KBY4	0,9621	CAAX amino terminal protease family	(Pfam) CAAX protease self-immunity
O53808	Q7D551	0,7145	ATP-dependent RNA helicase DEAD/DEAH box family	
Q8VKE5	A7S8F9	0,6342	DNA-directed RNA polymerase	
Q8VKE1	A4KFA3	0,8377	ATPase	

Capítulo 5

Discussão

O PSI-BLAST é uma das principais ferramentas para busca de homólogos distantes, conforme mostrado por (Mullera et al. (1999) e Anand et al. (2005)). Também é muito discutida na literatura a confiabilidade dos resultados encontrados pelo PSI-BLAST (Lee et al. (2008)). Conforme dissemos na introdução (seção 1.4.3), a maneira não supervisionada como a PSSM é calculada permite a incorporação de proteínas não homólogas que corrompem a matriz. Essa corrupção leva a cada vez mais incorporação de proteínas não relacionadas. Os nossos resultados mostraram que esse problema pode chegar a tal ponto que a pesquisa pode recusar a própria *query*. Em outras palavras, a *query* não se considera homologa a si mesma. Neste caso, é patente que os resultados não são confiáveis.

Um dos principais usos deste tipo de busca por homologo distante é na modelagem computacional (Combet et al. (2002)). Nessa técnica, uma proteína homóloga com estrutura terciária conhecida é usada como molde em um programa que "prediz" a estrutura 3D da *query*. Muitas vezes a *query* não possui nenhum homólogo próximo, que seria encontrado pelo BLASTp, e uma busca por homólogo distante precisa ser feita com PSI-BLAST. Embora esta busca seja em geral inspecionada por um curador manual, que verifica se o perfil das proteínas encontradas difere, não é plenamente conhecido pela comunidade bioinformática que a *query* pode desaparecer da busca.

Com esse problema em mente, os primeiros conceitos criados neste trabalho foram a auto pontuação e a auto pontuação relativa. Desta maneira, a pontuação da *query* é monitorada ao longo das iterações. Esse monitoramento permite detectar não só quando a *query* é excluída da busca, mas também ver se a *query* atribui uma pontuação baixa a si mesma. Neste caso, novamente a busca estaria considerando a *query* como não homologa, ou pouco homóloga a si mesma.

Em uma abordagem anterior de nosso trabalho, a filtragem de falso-positivo e verdadeiro positivo se deu por limite da auto-pontuação relativa. Um limite de 70% (0.7) de auto-pontuação relativa foi definido e resultados de iterações onde esta pontuação foi menor que 70% eram recusados. Esta abordagem foi empregada em um trabalho, (Guedes e colaboradores, não publicado). Neste trabalho, no entanto, a busca PSI-BLAST era feita em ambiente mais controlado (a comparação da query é feita com proteínas pré-selecionadas por outros agentes) diferente do que apresentamos nesta tese, em que as buscas são feitas no UniProt inteiro.

Um conceito que foi criado neste trabalho foi a pontuação relativa. Esta pontuação é obtida pela normalização da pontuação do *subject* pela auto pontuação (a princípio da *query*). Essa normalização torna as pontuações de diferentes BLASTs comparáveis entre si. Embora o *e-value*

seja um valor que se pode usar para fazer comparações entre BLASTs, a pontuação relativa foi proposta como forma alternativa. Em uma abordagem anterior esta pontuação era uma parte importante do que se reportava. Na atual abordagem essas pontuações se transformaram em atributos para os modelos de aprendizado de máquina.

No caso do BLASTp, tanto a pontuação relativa quanto o *e-value* podem ser usadas para se fazer comparações entre BLASTs. Para o PSI-BLAST, no entanto, a situação é bem diferente. Em primeiro lugar cada *subject* pode receber múltiplas pontuações e em segundo lugar essas pontuações (exceto da primeira iteração) são calculadas com uma PSSM que é construída de maneira não supervisionada e sem controle direto do usuário (Altschul et al. (1997)). Uma abordagem a estes problemas sugerida na literatura foi proposta por Lee et al. (2008) que combinou os resultados da primeira iteração, que é a menos corrompida, com os da última iteração que tem a maior taxa de revocação. Com esta combinação, os autores foram capazes de, em primeiro lugar, gerar uma pontuação única para o *subject*. Em segundo lugar, esta pontuação se mostrou eficiente em ordenar os *subjects* do mais provável para o menos provável.

A técnica proposta nesse trabalho se parece em auto nível com o que fizemos, uma vez que a probabilidade calculada por nosso modelo é uma combinação das pontuações dos diversas iterações. No entanto, diferente deste trabalho em que a combinação é linear, a nossa combinação é não linear. Em nosso modelo utilizamos a pontuação relativa de cada iteração bem como a auto pontuação relativa de cada iteração para calcular a probabilidade do *subject* ser do mesmo grupo da *query*, ou não. Mas assim como no trabalho de Lee et al. (2008), nosso modelo, é claro, ordenou os resultados de maneira superior ao *e-value* do PSI-BLAST.

Um ponto que é geralmente considerado ao se analisar um alinhamento de BLAST é a porcentagem do tamanho da *query* e do tamanho do *subject* que se alinharam. A pontuação normalizada pela auto pontuação da *query* no entanto pode esconder este fato. Em outras palavras, uma pontuação relativa alta pode ser obtida do alinhamento de uma *query* pequena com um *subject* grande. Um modelo eficiente de agrupamento deve levar isso em conta. Desta forma, criamos um segundo modo de normalização de pontuação, a relativização pela auto pontuação de *subject*. Neste caso, o alinhamento de uma *query* pequena com um *subject* grande pode ter pontuação normalizada pela auto-pontuação da *query* alta, mas a pontuação relativizada pela auto-pontuação do *subject* será baixa. Diferente de alguns trabalhos que filtram resultados com recobrimento baixo (Kaushik and at al. (2013)), nós não fizemos nenhuma pressuposição. Essas pontuações foram simplesmente passadas para os algoritmos de aprendizado de máquina e permitimos que eles aprendessem com os dados.

Conforme nós discutimos anteriormente, a auto-pontuação relativa é uma estimativa do quão deteriorada a PSSM está e, portanto, de quão confiáveis os resultados daquela iteração são. Desta forma, as pontuações relativas também foram incluídas no modelo. Novamente, nenhuma pressuposição foi feita e deixamos que o modelo aprendesse com os dados. Os resultados do gráfico 4.8, no entanto, mostram que a inclusão destes valores melhoraram a capacidade preditiva de nosso modelo. Isso mostra que a auto-pontuação realmente estima a confiabilidade da iteração. Caso contrário, sua informação seria ruído ou uma combinação linear da informação contida nas pontuações relativas. O uso da auto-pontuação é um outro ponto em que nossas técnicas se diferenciam daquelas descritas por Lee et al. (2008). Neste

trabalho não existe nenhuma forma de se estimar a deterioração da PSSM.

Em nossa revisão sobre uso de aprendizado de máquina em bioinformática encontramos vários trabalhos que reportam o uso de rede neural e SVM para diversos propósitos. O que não encontramos foi o uso de *ensemble* de modelos como nós fizemos. Embora não tenhamos feito uma busca direcionada para *ensemble* de modelos, o fato de não termos encontrado nenhum trabalho com esta técnica mostra que ela ainda é incomum na comunidade de bioinformática.

No trabalho de Gonzalez and Pearson (2010) foi feito um estudo sobre erros de PSI-BLAST e os autores reportam dois tipos de erros. No primeiro, proteínas não homologas são alinhadas com a *query*. No entanto, estes alinhamentos, de acordo com o autor, tinham *e-value* na faixa de 1e-3 a 1e-7 e os autores argumentam que tornar o *h-value* (inclusion_ethresh) mais estringente resolveria o problema. Em nosso caso, isso parece não ser um problema, já que usamos *h-value* de 1e-10. O outro tipo de erro que o autor reporta são alinhamentos que começam em uma região homóloga e estendem para região não homóloga, o que ajuda a corromper a PSSM. Esses alinhamentos, por usa vez, podem ter *e-values* tão baixos quanto 1e-70 tornando impraticável filtrá-los com *e-value*. Para nosso trabalho, no entanto, estes *subjects* não são falsos positivos pois eles de fato possuem regiões homólogas e mesmo no trabalho de Gonzalez and Pearson (2010) eles criaram um método para restringir o alinhamento e não para recusar estes *subjects*. Quanto à corrupção da matriz nosso modelo se mostrou eficiente em dar probabilidade baixa para proteínas que aparecem apenas quando a PSSM já esta corrompida.

Antes de movermos para o próximo tópico desta discussão uma ressalva se faz necessária. O compartilhamento de um motivo implica em ancestralidade comum? A resposta para esta pergunta é que pelo menos em alguns casos sim. Basta notar que uma proteína pode estar presente em mais de um KO ou PANTHER devido à presença de múltiplos domínios. Nosso modelo, no entanto, foi treinado em um conjunto de dados (PANTHER) que possibilita aos algoritmos aprender a identificar e classificar corretamente a homologia de domínios similares.

Alguns trabalhos como Kaushik and at al. (2013) reportam que o PSI-BLAST com uma única *query* não é capaz de recuperar toda a família da *query*. Estas observações estão de acordo com a nossa experiência com PSI-BLAST. Não obstante, isso não é problema para nós porque o objetivo deste trabalho é recuperar homólogos distantes, possivelmente para anotar proteínas hipotéticas (propagar a anotação de/para a query/seed) e não o de recuperar todo o seu *cluster*. Dito isso, nada impede que as técnicas de uso de múltiplas queries como a descrita por Kaushik and at al. (2013) sejam usadas com Annothetic.

Nos experimentos com quatro micro-organismos, testamos a eficiência do modelo em uma situação realística. Ficou evidenciado que mesmo em organismos considerados bem anotados fomos capazes de encontrar proteínas que não possuíam anotação e que nosso modelo foi capaz de sugerir anotação com alta probabilidade. Embora esse experimento tenha sido mais no sentido de aplicar a ferramenta que testá-la, a concordância das nossas sugestões com aquelas feitas por outros bancos de dados sugere uma forma de validação. Também houve anotações que nosso modelo propôs e que não foram sugeridas por outros bancos de dados como eggNOG (Muller and at al. (2010)) e Pfam (Bateman and et al (2002)), o que é importante para mostrar que nosso trabalho faz sugestões de anotação que os bancos de dados tradicionais não fazem.

Nos experimentos com proteínas recentemente anotadas, ficou evidenciado que das 900

proteínas testadas, 180 só tiveram alguma sugestão com PSI-BLAST. E estes experimentos também mostraram que existe uma correlação entre a probabilidade calculada pelo modelo e similaridade entre a anotação encontrada e a anotação da *query*. Esses dados comprovam a eficácia do modelo na anotação de proteínas hipotéticas. A forma de comparação de anotações que nós utilizamos tende, no entanto, a superestimar erros, porque não utilizamos uma lista de sinônimos.

Além de um simples experimento de previsão de anotação, nós fomos capazes de extrair uma nova estatística desse experimento. Neste caso, treinamos um modelo linear (polinomial) para ponderar a importância de cada anotação e criar um consenso. Em seguida, pudemos ordenar as anotações pela sua concordância com este consenso. Essa técnica teve sua eficácia e é baseada em princípios semelhantes aos que regem o *ensemble*. A predição de muitos é melhor que a predição de um só. Neste caso, se muitas descrições confirmam uma anotação, a confiança nesta anotação, naturalmente, cresce. Isso é reforçado pelo fato de que anotações dos *subjects* incorretos tendem a não formar um consenso, ou seja, estes *subjects* têm cada um uma anotação diferente. Esta abordagem é diferente daquela usada em muitos trabalhos de bioinformática, como Moriya and at al. (2007) e Barbosa-Silva et al. (2008) que usam o *best-hit*, e deve ser empregada com cuidado, pois, para proteínas pertencentes a uma família pequena, o consenso pode acabar sendo corrompido. Mas nossa ferramenta não se baseia apenas neste consenso e o *best-hit* pode ser usado. Além do Mais, este consenso pode ser criado usando apenas proteínas com 'alta' probabilidade.

O Annothetic, além de calcular a confiança e o nível de concordância com o consenso, também calcula a distância filogenética do *subject* com a *query*. Isto é feito com dados de um alinhador múltiplo (MUSCLE). Essa distância, no entanto, só é calculada para os primeiros 300 alinhamentos, ordenados pela confiança, pois é um cálculo custoso. O *E-value* e a identidade da primeira iteração em que o *subject* aparece também são informadas. Isso ajuda o usuário a manter contato com as velhas métricas, que embora inferiores, são mais bem compreendidas. O tamanho do alinhamento pode ser usado para se saber se se trata de um domínio homólogo ou proteína inteira. O LCA-level e LCA são outras duas métricas reportadas para cada resultado. Esses valores mostram o quão longe na árvore da vida os *subjects* estão. Por fim, a auto-pontuação e a iteração do *subject* são mostrados. Todos estes valores formam um relatório bem completo das proteínas que o Annothetic encontra.

Capítulo 6

Conclusões

A partir do que foi exposto neste trabalho concluímos que o uso de PSI-BLAST é uma forma eficaz de busca por homólogos distantes. A associação dessa ferramenta com modelos de aprendizado de máquina supervisionado é capaz de solucionar os principais problemas do PSI-BLAST que são: sua alta taxa de falsos positivos com seu *e-value* e pontuação de interpretações não tão simples como os equivalentes em BLASTp. Também ficou evidente que o uso de *ensemble* de modelos, ao invés de modelos individuais, é a melhor forma de se trabalhar e esta estratégia, que já é comum em outras áreas da computação, deverá ser mais adotada em bioinformática.

A busca por homólogos distantes pode ser útil a variados propósitos, mas neste trabalho demonstramos seu poder como modo de sugerir anotação para proteínas hipotéticas.

Também é uma conclusão deste trabalho que a busca de anotação utilizando-se de todas proteínas encontradas ao invés de apenas o *best-hit* é uma maneira promissora, embora seja necessário encarar este método com o devido cuidado.

Concluímos a partir dos resultados dos quatro micro-organismos que mesmo em organismos bem anotados existe um grande número de proteínas que permanecem sem função conhecida e que podem tirar proveito de nossos modelos. Muitas destas proteínas inclusive não recebem qualquer sugestão de anotação com BLASTp mas recebem sugestões com PSI-BLAST, sendo que, para muitas destas sugestões, foram calculadas probabilidades relativamente altas pelos nossos modelos.

Acrescentamos ainda, que a construção de ferramentas que calculam e reportam diversas métricas é importante para se criar um relatório mais completo. Este tipo de relatório pode ser útil a diversos perfis de usuários, além de apresentar resultados mais fáceis de se avaliar em relação à sua confiabilidade.

Por fim, este trabalho gerou a ferramenta que implementa todos esses modelos. Essa ferramenta já está disponível em sua versão web e esperamos que, em breve, também esteja disponível na versão desktop.

Como perspectivas deste trabalho temos vários refinamentos que podem ser feitos ao modelo de aprendizado de máquina como: treinar um modelo que seja otimizado para tamanho da *query* ou dos *clusteres*. Também temos refinamentos que podem ser feitos à parte de mineração de anotação como o uso de lista de sinônimos e a incorporação de ontologia.

Alem dos refinamentos dos modelos outras perspectivas que este trabalho deixa é a aplicação de nossos modelos em novos grupos de proteínas hipotéticas e em novos genomas além dos quatro usados neste trabalho.

Como uma ultima perspectiva que deixamos esta a expansão de nosso modelo com o uso de estratégias que tiram proveito das relações transitivas de homologia, como a citada na introdução 1.3.

Referências Bibliográficas

Altschul, S., Madden, T., Schäffer, A., Zhang, J., Zhang, Z., Miller, W., and Lipman, D. (1997). Gapped blast and psi-blast: a new generation of protein database search programs. *Nucleic Acids Res*, 25(17):3389–402.

Altschul, S. F. e. a. (1990). Basic local alignment search tool. *J Mol Biol*, 215(3):403–410.

Anand, B., Gowri, V., and Srinivasan, N. (2005). Use of multiple profiles corresponding to a sequence alignment enables effective detection of remote homologues. *bioinformatics*, 21(12):2821–2826.

Apweiler, R. and et al (2004). Uniprot: the universal protein knowledgebase. *Nucleic Acids Res*, (Database issue):D115–9.

Avriel, M. (2003). *Nonlinear Programming: Analysis and Methods*. Dover Publishing, ISBN 0-486-43227-0.

Barbosa-Silva, A., Satagopam, V. P., Schneider, R., and Ortega, J. M. (2008). Clustering of cognate proteins among distinct proteomes derived from multiple links to a single seed sequence. *BMC Bioinformatics*, 9(141).

Bateman, A. and et al (2002). The pfam protein families database. *Nucleic Acids Res*, 30(1).

Berman, H. M. (2008). The protein data bank: a historical perspective. *Acta Crystallographica Section A*, pages 88–95.

Bettella, F., Rasinski, D., and Knapp, E. W. (2012). Protein secondary structure prediction with sparrow. *J. Chem. Inf. Mode*, 52(2):545–556.

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. Springer.

Bornholdt, S. and Graudenz, D. (1992). General asymmetric neural networks and structure design by genetic algorithms. *Neural Networks*, 5:327–334.

Breiman, L. (1996). Bagging predictors. *Machine Learning*.

Breiman, L. (2001). Random forests. *Machine Learning*.

Catral, M. and at al. (2004). On reduced rank nonnegative matrix factorization for symmetric nonnegative matrices. *Linear Algebra and its Applications*, 393:107–126.

Chatterjee, P., Basu, S., Kundu, M., Nasipuri, M., and Plewczynski, D. (2011). Psp_mcsvm: brainstorming consensus prediction of protein secondary structures using two-stage multiclass support vector machines. *J. Mol. Model.*, 17:2191–2201.

Chauhan, J. S., Mishra, N. K., and Raghava, G. P. (2009). Identification of atp binding residues of a protein from its primary sequence. *BMC Bioinformatics*, 10(434).

Collobert, R. and Bengio, S. (2004). Links between perceptrons, mlps and svms. *Proc. Int'l Conf. on Machine Learning (ICML)*.

Combet, C., Jambon, M., Deleage, G., and Geourjon, C. (2002). Geno3d: automatic comparative molecular modelling of protein. *Bioinformatics*, 18(1):213–214.

Consortium, U. (2009). The universal protein resource (uniprot). *Nucleic Acids Res*, 37(Database issue):D169–74.

Dayhoff, M. (1978). Atlas of protein sequence and structure. *National Biomedical Research Foundation, Washington*.

Dietterich, T. G. (2000). Ensemble methods in machine learning. *Multiple Classifier Systems: Lecture Notes in Computer Science*, 1857:1–15.

D.Thomas, P. and at al (2003). Panther: a browsable database of gene products organized by biological function, using curated protein family and subfamily classification. *Nucleic Acids Research*, 31(1).

Edgar, R. (2004). Muscle: multiple sequence alignment with high accuracy and high throughput. *Nucleic Acids Res*, 32(5):1792–1797.

Fabris, F., Sgarro, A., and Tossi, A. (2007). Splitting the blosum score into numbers of biological significance. *Journal on Bioinformatics and Systems Biology*.

Felsenstein, J. (2002). Phylip (phylogeny inference package) version 3.6a3. *Department of Genome Sciences, University of Washington, Seattle*.

Garg, A. and Gupta, D. (2008). Virulentpred: a svm based prediction method for virulent proteins in bacterial pathogens. *BMC Bioinformatics*, 9(62).

Ghanty, P. and at al (2013). Prediction of protein secondary structure using probability based features and a hybrid system. *Journal of Bioinformatics and Computational Biology*, 11(5).

Gonzalez, M. W. and Pearson, W. R. (2010). Homologous over-extension: a challenge for iterative similarity searches. *Nucleic Acids Research*, 38(7):2177–2189.

Gribskov, M. and at al (1987). Profile analysis: detection of distantly related proteins. *Proc. Natl. Acad. Sci. USA*, 84:4355 – 4358.

Henikoff, S. and Henikoff, J. G. (1991). Automated assembly of protein blocks for database searching. *Nucl. Acids Res*, 19(23):6565–6572.

Henikoff, S. and Henikoff, J. G. (1992). Amino acid substitution matrices from protein blocks. *Proc Natl Acad Sci U S A*, 89(22):10915–10919.

Hochreiter, S., Heusel, M., and Obermayer, K. (2007). Fast model-based protein homology detection without alignment. *Bioinformatics*, 23(14):1728–1736.

Hornik, K., Stinchcombe, M., and White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366.

Johnson, L. S., Eddy, S. R., and Portugaly, E. (2010). Hidden markov model speed heuristic and iterative hmm search procedure. *BMC Bioinformatics*, 11(431).

Jones, D. T. and Swindells, M. B. (2002). Getting the most from psi-blast. *Trends in Biochemical Sciences*, 27(3):161–164.

Kabsch, W. and Sander, C. (1983). Dictionary of protein secondary structure: Pattern recognition of hydrogen-bonded and geometrical features. biopolymers. *Biopolymers*, 22(12).

Kalita, M. K. and at al (2008). Cyclinpred: A svm-based method for predicting cyclin protein sequences. *Plos One*, 3(7).

Kanehisa, M. (1997). A database for post-genome analysis. *Trends Genet*, 13(9):375–376.

Kanehisa, M. and et al (2004). The kegg resource for deciphering the genome. *Nucleic Acids Res*, 32(Database issue):D277–80.

Karlin, S. and Altschul, S. (1990). Methods for assessing the statistical significance of molecular sequence features by using general scoring schemes. *Proc Natl Acad Sci U S A*, 87:2264–2268.

Kaushik, S. and at al. (2013). Improved detection of remote homologues using cascade psi-blast: Influence of neighbouring protein families on sequence coverage. *Plos One*, 8(2).

Klema, V. and Laub, A. (1980). The singular value decomposition: Its computation and some applications. *Automatic Control*, 25(2):164 – 176.

Krogh, A., Brown, M., Mian, I. S., Sjolander, K., and Haussler, D. (1994). Hidden markov model in computational biology: Applications to protein modeling. *J. Mol. Biol.*

Krogh, A. and Sollich, P. (1996). Learning with ensembles: How over-fitting can be useful. *Advances in Neural Information Processing Systems 8*, pages 190–196.

Kumar, M. and at al (2011). Svm based prediction of rna-binding proteins using binding residues and evolutionary information. *Journal of Molecular Recognition*, 24(2):303–313.

Kumar, R., Panwar, B., Chauhan, J. S., and Raghava, G. P. (2011). Analysis and prediction of cancerlectins using evolutionary and domain information. *BMC Research Notes*, 4(237).

Larkin, M. A. and et al (2007). Clustal w and clustal x version 2.0. *Bioinformatics*, 23(21):2947–8.

Lee, M. M., Chan, M. K., and Bundschuh, R. (2008). Simple is beautiful: a straightforward approach to improve the delineation of true and false positives in psi-blast searches. *Bioinformatics*, 24(11):1339–1343.

Liu, B., Wang, X., Lin, L., Dong, Q., and Wang, X. (2008). A discriminative method for protein remote homology detection and fold recognition combining top-n-grams and latent semantic analysis. *BMC Bioinformatics*, 9(510):1–16.

Liu, T., Genga, X., Zheng, X., Li, R., and Wang, J. (2012). Accurate prediction of protein structural class using auto covariance transformation of psi-blast profiles. *Journal*, 42(6):2243–2249.

Loytynoja, A. and Goldman, N. (2010). webprank: a phylogeny-aware multiple sequence aligner with interactive alignment browser. *BMC Bioinformatics*, 11(509):3 – 7.

Mason, J., Shepherd, M., and Duffy, J. (2009). An n-gram based approach to automatically identifying web page genre. *System Sciences*, pages 1–10.

Meira, W. and Zaki, M. J. (2013). *Data Mining and Analysis: Fundamental Concepts and Algorithms*. Cambridge University Press.

Melvin, I. and at al (2007). Svm-fold: a tool for discriminative multi-class protein fold and superfamily recognition. *BMC Bioinformatics*, 8(Suppl 4).

Mi, H. and at al (2005). The panther database of protein families, subfamilies, functions and pathways. *Nucleic Acids Research*, 33(Database issue).

Mi, H. and at al (2010). Panther version 7: improved phylogenetic trees, orthologs and collaboration with the gene ontology consortium. *Nucleic Acids Research*, 38(Database issue).

Mi, H., Guo, N., Kejariwal, A., and Thomas, P. D. (2007). Panther version 6: protein sequence and function evolution data with expanded representation of biological pathways. *Nucleic Acids Research*, 35(Database issue).

Mitchell, T. (1997). *Machine Learning*. McGraw Hill, ISBN 0-07-042807-7.

Moriya, Y. and at al. (2007). Kaas: an automatic genome annotation and pathway reconstruction server. *Nucl. Acids Res.*, 35(suppl 2):W182–W185.

Moult, J. (1999). Predicting protein three-dimensional structure. *Curr. Opin. Biotechnol*, 10:583 – 588.

Muller, J. and at al. (2010). eggnog v2.0: extending the evolutionary genealogy of genes with enhanced non-supervised orthologous groups, species and functional annotations. *Nucl. Acids Res.*, 38(suppl 1):D190–D195.

Mullera, A., MacCalluma, R. M., and Sternberg, M. J. (1999). Benchmarking psi-blast in genome annotation. *Journal of Molecular Biology*, 293(5):1257–1271.

Needleman, S. B. and Wunsch, C. D. (1970). A general method applicable to the search for similarities in the amino acid sequence of two proteins. *J Mol Biol*, 48(3):442–453.

Opitz, D. and Maclin, R. (1999). Popular ensemble methods: An empirical study. *Journal of Artificial Intelligence Research*, 11:69–198.

Oua, Y.-Y., Gromihab, M., Chena, S.-A., and Suwab, M. (2008). Tmbetadisc-rbf: Discrimination of beta-barrel membrane proteins using rbf networks and pssm profiles. *Computational Biology and Chemistry*, 32(3):227–231.

Paice, C. D. (1996). Method for evaluation of stemming algorithms based on error counting. *Journal of the American Society for Information Science*, 47(8):632–649.

Price, M., Dehal, P., and Arkin, A. (2010). Fasttree 2 - approximately maximum-likelihood trees for large alignments. *PLoS ONE*, 5(3).

Rashid, M., Saha, S., and Raghava, G. P. (2007). Support vector machine-based method for predicting subcellular localization of mycobacterial proteins using evolutionary information and motifs. *BMC Bioinformatics*, 8(337).

Repsys, V., Margelevicius, M., and Venclovas, C. (2008). Re-searcher: a system for recurrent detection of homologous protein sequences. *BMC Bioinformatics*, 9(296).

Rost, B., Sander, C., and Schneider, R. (1994). Redefining the goals of protein secondary structure prediction. *J. Mol. Biol.*

Rychlewski, L. and at al (2000). Comparision of sequence profiles. strategies for structural predictions using sequence information. *Protein Sci.*, 9:232 – 241.

Salamov, A. and at al (1999). Combining sensitive database searchs with multiple intermediates to detect distant homologues. *Protein Eng.*, 12:95 – 100.

Schaffer, A. A. and at al (1999). Impala: matching a protein sequence against a collection of psi-blast-constructed position-specific score matrices. *Bioinformatics*, 15(12).

Schäffer, A. A. and et al (2000). Improving the accuracy of psi-blast protein database searches with composition-based statistics and other refinements. *Nucl. Acids Res*, 29(14):2994–3005.

Shah, A. R., Oehmen, C. S., and Webb-Robertson, B.-J. (2008). Svm-hustle-an iterative semi-supervised machine learning approach for pairwise protein remote homology detection. *Bioinformatics*, 24(6):783–790.

Shameer, K., Nagarajan, P., Gaurav, K., and Sowdhamini, R. (2009). 3pfdb - a database of best representative pssm profiles (brps) of protein families generated using a novel data mining approach. *BioData Mining*, 2(8).

Silva, L., Davis, A., and H, R. (2012). A feature engineering approach for click-through rate prediction: Kdd cup track 2. *avalable at https://kaggle2.blob.core.windows.net/competitions/kddcup2012/2748/media/BirutasTeam.pdf*.

Sim, J., Kim, S.-Y., and Lee, J. (2005). Pprodo: Prediction of protein domain boundaries using neural networks. *Proteins: Structure, Function, and Bioinformatics*, 59(3):627–632.

Simon, P. (2013). *Too Big to Ignore: The Business Case for Big Data*. Wiley, ISBN 978-1118638170.

Smith, T. F. and Waterman, M. S. (1981). Identification of common molecular subsequences. *J Mol Biol*, 147(1):195–197.

Stormo, G., Schneider, T. D., Gold, L., and Ehrenfeucht, A. (1982). Use of the 'perceptron' algorithm to distinguish translational initiation sites in e. coli. *Nucleic Acids Res*, 10(9):2997–3011.

Stroock, D. W. (2005). *An Introduction to Markov Processes*. Springer.

Tamura, K., Dudley, J., Nei, M., and Kumar, S. (2007). Mega4: Molecular evolutionary genetics analysis (mega) software version 4.0. *Mol Biol Evol*, 24(8):1596–1599.

Tetko, I. V., Livingstone, D. J., and Luik, A. I. (1995). Neural network studies. 1. comparison of overfitting and overtraining. *J. Chem. Inf. Comput. Sci*, 25:826–833.

Tomovica, A., Janicicb, P., and Keseljc, V. (2006). n-gram-based classification and unsupervised hierarchical clustering of genome sequences. *Computer Methods and Programs in Biomedicine*, 81(2):137–153.

Uehara, K., Kawabata, T., and Go, N. (2004). Filtering remote homologues using predicted structural information. *Protein Engineering, Design and Selection*, 17(7):565–570.

Wang, L., Huang, C., Yang, M. Q., and Yang, J. Y. (2010). Bindn+ for accurate prediction of dna and rna-binding residues from protein sequence features. *BMC Systems Biology*, 4(Suppl 1).

Weizhong, L. and at al (2000). Saturated blast: an automated multiple intermediate sequence search used to detect distant homology. *Bioinformatics*, 16(12):1105 – 1100.

Wilbur, W. J. and Sirotkin, K. (1992). The automatic identification of stop words. *Journal of Information Science*, 18(1):45–55.

Willett, P. (2006). The porter stemming algorithm: then and now. *Program: electronic library and information systems*, 40(6):219–223.

Xia, X. (2012). Position weight matrix, gibbs sampler, and the associated significance tests in motif characterisation and prediction. *Scientifica*, 2012.

Yan, R.-X., Si, J.-N., Wang, C., and Zhang, Z. (2009). Descfold: A web server for protein fold recognition. *BMC Bioinformatics*, 10(416).

Zemla, A., Venclovas, C., Fidelis, K., and Rost, B. (1999). A modified definition of sov, a segment-based measure for protein secondary structure prediction assessment. *Proteins: Structure, Function, and Bioinformatics*, 34(2):220–223.

Zou, L., Wang, Z., and Huang, J. (2007). Prediction of subcellular localization of eukaryotic proteins using position-specific profiles and neural network with weighted inputs. *Journal of Genetics and Genomics*, 34(12):1080–1087.

Apêndice A

Bio Stop Words

Tabela A.1: Lista das bio-stopWords definidas por nós neste trabalho. Estas palavras foram filtradas manualmente apartir de uma lista de 672 palavras que ocorreram mais de 10 mil vezes em todo o UniProt versão de fevereiro de 2012.

protein	uncharacterized	putative	polypeptide
fragment	family	subunit	binding
domain	regulator	precursor	predicted
like	type	factor	dependent
chain	beta	system	component
containing	related	specific	probable
amino	superfamily	polyprotein	conserved
and	large	of	associated
shotgun	whole	assembly	class
sequence	small	enzyme	major
repeat	hypothetical	directed	facilitator
complex	to	scaffold	terminal
region	undetermined	homolog	short
kda	similar	subfamily	with
group	tail	box	isoform
in	delta	export	mitochondrial
nonstructural	secreted	oligopeptide	non
possible	rich	member	inhibitor
the	had	structural	systems
independent	for	variant	involved
cluster	release	determining	exported
gene	heavy	exporter	import
element	motif	single	insertion
formation	plasmid	long	preprotein
recognition	interacting	high	mismatch
expressed	center	poly	pentapeptide
forming	regulated	specificity	reaction
effector	processing	minor	particle
highly			