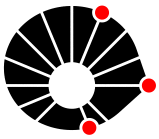

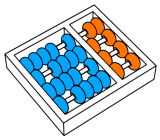

Predição de Funções Moleculares de Proteínas utilizando Aprendizado de Máquina

Felipe Lopes de Mello *Gabriel Bianchin de Oliveira Zanoni Dias*

Relatório Técnico - IC-PFG-21-44
Projeto Final de Graduação
2021 - Dezembro

UNIVERSIDADE ESTADUAL DE CAMPINAS
INSTITUTO DE COMPUTAÇÃO

Predição de Funções Moleculares de Proteínas utilizando Aprendizado de Máquina

Felipe Lopes de Mello Gabriel Bianchin de Oliveira Zanoni Dias

Resumo

Dados de Novembro de 2021 mostram que existem mais de 200 milhões de proteínas nas quais suas funções moleculares ainda são desconhecidas. Como a determinação empírica destas funções são lentas e custosas, diversos grupos de pesquisa ao redor do mundo tem aplicado aprendizado de máquina para realizar a predição de funções de proteínas. Neste trabalho, avaliamos o uso desde métodos clássicos até arquiteturas baseadas em Transformers. Nossa melhor arquitetura consiste no uso dos *embeddings* resultantes de dois Transformers (um pré-treinado com o ProtBERT e outro com o ProtBERT-BFD) como entrada para uma MLP. Este modelo obteve um F_{max} de 0,562 na nossa base de dados, e, ao aplicarmos este modelo na mesma base utilizada pelo DeepGOPlus, que é um dos principais métodos para realizar a predição de funções de proteínas, atingimos o valor de 0,617, superando o melhor resultado da literatura.

1 Introdução

As proteínas possuem um papel muito importante na biologia, sendo responsáveis desde funções estruturais até a catalisação de reações vitais para o metabolismo de seres vivos [34]. Na data da publicação deste trabalho, segundo dados da base de proteínas UniProtKB [32], de um total de mais de 225 milhões de proteínas menos de 0,3% possuem suas funções catalogadas. Mais do que isso, em muitas amostras presentes nesta base a única informação disponível é a sequência de aminoácidos. Para a determinação das funções, são necessários métodos bioquímicos capazes de determinar as funções moleculares das proteínas empiricamente, mas seu alto custo inviabiliza seu uso na prática em uma base tão grande.

Para a classificação de funções exercidas pelas proteínas, utiliza-se da Ontologia Genética (OG) proposta por Ashburner *et al.* [3], na qual as funções de proteína são estruturadas de forma hierárquica. Nesta ontologia, já existem mais de 40.000 termos distribuídos em 3 domínios: Ontologia de Função Molecular (OFM), Ontologia de Processo Biológico (OPB) e Ontologia de Componentes Celulares (OCC).

Diversos trabalhos já foram apresentados com o objetivo de propor as melhores soluções para este problema. Utilizando a base do CAFA3[1] [38] de 2019, Kulmanov e Hoehndorf conseguiram obter um F_{max} de 0,557 com o DeepGOPlus [16], um método que combina uma rede neural convolucional (CNN, do inglês *Convolutional Neural Network*) [18] com predições baseadas em similaridade de sequência. Este resultado superou seu próprio método anterior, o DeepGO [17], que ainda não possuía a informação de *motifs* na rede e ainda usava *embedding* ao invés de *one-hot encoding*, atingindo 0,470 na base de 2017.

Com a base de dados do Swiss-Prot de 2016, You *et al.* [36] alcançaram o valor de 0,631 com o DeepText2GO, um modelo que analisa o texto de trabalhos que citam a função da proteína e sua sequência através do BLAST [1] e prediz por meio de uma regressão logística [22] ou KNN [10]. Os

[1] `https://www.biofunctionprediction.org/cafa`

mesmos autores já tinham atingido o valor de 0,586 na mesma base de 2014 com o GOLabeler [37], um método que combina 5 classificadores diferentes para fazer a predição.

Em relação aos trabalhos que utilizaram processamento de linguagem natural para problemas biológicos, temos o UDSMProt [29] que utilizou camadas de LSTMs [14] para fazer a predição, obtendo resultados próximos aos modelos anteriores. Elnaggar et al. [9] propuseram arquiteturas baseadas em BERT [8] que foram pré-treinadas com proteínas. A partir destas arquiteturas, é possível realizar o ajuste fino para a tarefa desejada.

Neste trabalho, avaliamos diversos classificadores na ontologia de função molecular, visto que esta ontologia é a mais dependente da sequência de aminoácidos. Nós avaliamos diversos preditores, desde técnicas clássicas de aprendizado de máquina, como Random Forest, XGBoost e Regressão Logística, até modelos de aprendizado profundo, como redes convolucionais, redes recorrentes e Transformers.

Por último, desenvolvemos um método no qual os *embeddings* gerados pelos 2 melhores Transformers são usados como entrada para uma MLP (do inglês *Multilayer Perceptron*) [26]. Este método nos permitiu alcançar o valor de 0,562 na métrica de avaliação F_{max}. Além disso, executando este método na mesma base usada pelo DeepGOPlus, fomos capazes de superar seu resultado (o melhor até então nesta base) em 0,060.

O resto deste documento está organizado da seguinte forma. Na Seção 2, daremos a base teórica necessária para entender conceitos e métodos usados neste trabalho. Na Seção 3, apresentamos a base de dados utilizada e análises sobre os seus dados. Na Seção 4, explicamos como os métodos de aprendizado de máquina foram aplicados sobre esta base. Na Seção 5, discutimos os resultados obtidos por cada modelo testado. Na Seção 6, concluímos o trabalho e apontamos possíveis direções para trabalhos futuros.

2 Fundamentação Teórica

Vários conceitos e definições presentes neste trabalho requerem conhecimentos desde a área da biologia até técnicas de aprendizado de máquina. Nesta seção, descrevemos e explicamos brevemente cada conceito utilizado.

2.1 Proteína

Proteínas são macromoléculas presentes em todas as células dos seres vivos, possuindo um papel importante no fornecimento de estrutura para células e organismos, catalisação de reações metabólicas, transporte de moléculas e replicação de DNA. As proteínas são compostas por sequências de aminoácidos com tamanho variável. Conhecemos a existência de cerca de 500 aminoácidos presentes na natureza, mas apenas 20 são encontrados no código genético [11]. Como uma proteína pode possuir, em teoria, um número infinito de aminácidos, também temos um número infinito de proteínas possíveis.

Os aminoácidos, por sua vez, são compostos orgânicos que contém amina, ácido carboxílico e uma cadeia de átomos de carbono ou hidrogênio (chamada grupo-R) que varia de acordo com cada aminoácido [19]. Nas bases de proteínas públicas [32], os aminoácidos são representados como letras do alfabeto, uma vez que na maioria dos estudos científicos o principal interesse se limita às proteínas encontradas no corpo humano e, portanto, temos apenas 20 tipos de aminoácidos. Além disso, em algumas bases de dados, temos uma representação múltipla de alguns aminoácidos, como a Alanina, que é representada pelas letras “A” e “X”, Asparagina, que é representada pelas letras “N” e “B”, e Glutamina, que é representada pelas letras “Q” e “Z”.

2.2 Ontologia Genética

Como consequência do número exponencial de proteínas, com o tempo também foram encontrados um grande número de funções distintas que cada uma delas possui. Isso fez com que se fosse necessário encontrar uma maneira de organizar todas as funções de proteína de uma forma clara e objetiva. O procedimento adotado pela literatura científica foi a ontologia genética (GO, do inglês *Gene Ontology*). Nesta ontologia, todas as funções são dividas em 3 domínios: Ontologia de Função Molecular (OFM), Ontologia de Processo Biológico (OPB) e Ontologia de Componentes Celulares (OCC), como mostrado na Figura 1. Assim, cada função proteica é associada a um termo GO e, por isso, neste trabalho usamos os conceitos "função" e "termo" como sinônimos.

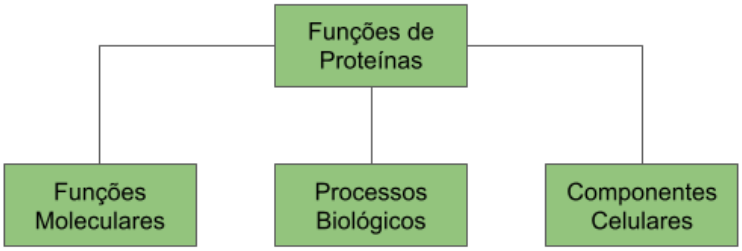

Figura 1: Ontologia genética com os seus 3 domínios: Funções Moleculares, Processos Biológicos e Componentes Celulares.

Na ontologia de função molecular, temos as funções responsáveis por atividades elementares a níveis moleculares. Um exemplo de função é a atividade catalítica (que é associada ao termo GO:0003674) e possui o papel de catálise (aumento da velocidade de uma reação bioquímica) em temperaturas fisiológicas. Esta ontologia é a que apresenta maior correlação com a sequência de aminoácidos quando comparado aos demais domínios [2]. Esta é uma característica importante, uma vez que em muitas bases de dados a única informação presente das amostras é a sequência de aminoácidos.

As operações pertinentes ao funcionamento de unidades vivas como células, tecidos, órgãos e organismos estão no conjunto de processos biológicos. Um exemplo de termo nesta ontologia é o da contração do coração (GO:0060047) que é a operação responsável por diminuir o volume do coração, impulsionando assim o sangue para o resto do corpo.

No último domínio, componentes celulares, temos as funções de composição de partes da célula ou do ambiente extracelular. Como exemplo, podemos mencionar a manutenção de junção celular (GO:0034331), responsável por preservar a região da conexão entre duas células ou de uma célula com seu ambiente extracelular.

Ainda na ontologia genética, além de existirem esses três domínios, cada função pode ter outras funções como *filhos*, sendo termos mais específicos que os pais, formando, assim, um Grafo Acíclico Dirigido (DAG, do inglês *Directed Acyclic Graph*). Além disso, cada proteína pode ter mais de uma função e estas funções não precisam necessariamente pertencer ao mesmo domínio. Para demonstrar este comportamento, ilustramos a árvore que compõe a função de contração do coração na Figura 2. Esta hierarquia nos auxilia na predição dos nós pais, uma vez que afirmar que um termo de um nó filho está presente em uma proteína implica que podemos afirmar que o nó pai também está presente.

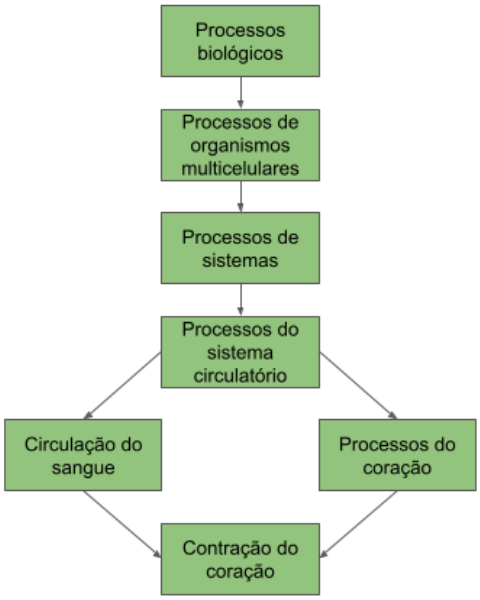

Figura 2: Exemplo do DAG atrelado à ontologia genética "contração do coração" (GO:0060047).

2.3 Aprendizado de Máquina

Nesta seção, introduzimos alguns modelos de aprendizado de máquina que foram utilizados neste trabalho. Aprendizado de máquina se refere à qualquer algoritmo que consegue realizar tarefas sem que ele seja especificamente programado para isso, sendo capaz de melhorar sua solução através da experiência obtida pelos dados fornecidos [23].

2.3.1 Regressão Logística

A regressão logística é um método que foi inicialmente usado pela estatística com o objetivo de predizer a probabilidade de uma classe existir ou um evento ocorrer [13]. Neste trabalho, focaremos na regressão logística binária, ou seja, existem apenas dois resultados possíveis: a classe estar presente (o que representamos como 1) ou não estar presente (representado por 0).

Neste modelo, as possibilidades são descritas através da função logística dada pela Equação 1, na qual x_0 é o valor de x no ponto médio da curva sigmoide, L é o valor máximo da curva e k é a declividade da curva. Naturalmente, esta equação descreve o caso de apenas uma classe por simplificação. Ao encontrar a melhor curva que descreve o comportamento das amostras de treino, somos capazes de predizer a probabilidade de um dado no conjunto de teste.

$$f(x) = \frac{L}{1 + e^{-k(x-x_0)}} \tag{1}$$

2.3.2 Random Forest

Uma árvore de decisão é um modelo de classificação no qual uma árvore é construída a partir dos dados de treino e as predições são realizadas por meio de decisões do tipo "se, senão". Nesta árvore, as folhas representam os rótulos de cada classe, os nós são as avaliações de características presentes na entrada e os ramos representam as decisões possíveis que podem ser tomadas com base nessa

avaliação [25]. Assim, ao fornecemos uma sequência de aminoácidos, a árvore decide em cada nó qual caminho deve ser percorrido até que uma folha seja alcançada.

Para a construção da árvore de decisão, começamos com o nó raiz que possui o conjunto das amostras de treino. Depois, dividimos essas amostras em subconjuntos de acordo com um critério, e cada subconjunto se torna um novo nó. Os demais nós da árvore são construídos recursivamente, repetindo o mesmo processo descrito para a raiz.

O Random Forest é um método que utiliza uma floresta de árvores de decisão para realizar a classificação onde cada árvore recebe como entrada um subconjunto aleatório das amostras de treino [31]. Nessa floresta, como cada árvore pode realizar uma predição diferente, o objetivo se torna então adotar um método de junção (*ensemble*) das predições de cada árvore.

2.3.3 XGBoost

O XGBoost é um método que une a árvore de decisão com uma técnica chamada *gradient boosting* [4]. A ideia desta técnica é otimizar a função de custo utilizada pelo classificador, tornando, assim, a classificação mais eficiente.

2.4 Aprendizado Profundo

Embora as técnicas de aprendizado profundo estejam dentro do conjunto dos métodos de aprendizado de máquina, optamos por separar nesta seção os modelos baseados em redes neurais.

2.4.1 *Multilayer Perceptron*

O *Multilayer Perceptron* (MLP) é uma rede neural que consiste de pelo menos 3 camadas: a camada de entrada, uma ou mais camadas ocultas, e uma camada de saída. Cada camada pode possuir mais de um neurônio, no qual, com exceção da camada de entrada, usa uma função de ativação para determinar sua saída conforme a entrada e também possui conexão com todos os neurônios da camada anterior e da camada seguinte. Além disso, o MLP utiliza a técnica *backpropagation* de aprendizado supervisionado, responsável por calcular o gradiente da função de perda e ajustar os pesos da rede. Um exemplo desse tipo de rede é ilustrado na Figura 3.

2.4.2 Rede Neural Convolucional

Uma rede neural convolucional (CNN) é uma adaptação da MLP, apresentada anteriormente, na qual, pelo menos, 2 tipos de camadas diferentes (convolução e *pooling*) são introduzidas antes da entrada da MLP. Um exemplo de rede neural convolucional é a Inception-V4 [30].

Na CNN, temos na primeira camada a camada de convolução que aplica um filtro nos dados da entrada com o objetivo de criar um vetor (ou matriz) capaz de resumir as características detectadas na entrada. Após esta etapa, o vetor de características resultante passa por uma camada de *pooling*, que é responsável por reduzir a dimensionalidade dos dados, diminuindo assim a complexidade computacional necessária para o treinamento do modelo. Finalmente, os resultados são usados como entrada na MLP, que realiza a classificação.

2.4.3 Rede Neural Recorrente Bidirecional

Redes neurais recorrentes (RNN) possuem a capacidade de lidar com dados sequenciais. Os neurônios utilizados nesta arquitetura possuem ligações entre as unidades de camada. Dentre

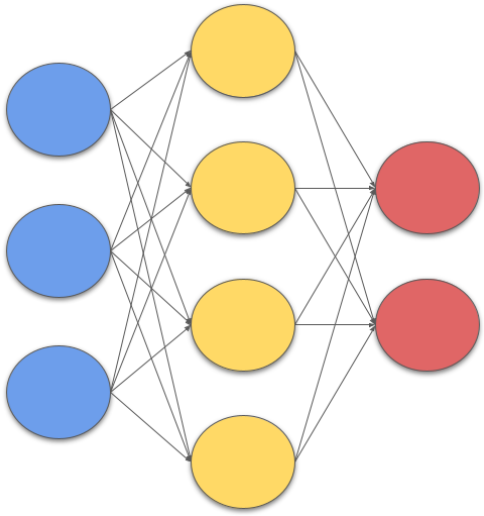

Figura 3: Exemplo de uma MLP com a camada de entrada com 3 neurônios (em azul), uma camada oculta com 4 neurônios (em amarelo) e a camada de saída com 2 neurônios (em vermelho).

os neurônios mais utilizados estão o LSTM (do inglês *Long Short-Term Memory*) [14] e o GRU (do inglês *Gated Recurrent Unit*) [5].

Devido às limitações da arquitetura padrão para problemas que dependem também de dados futuros, ou dados posteriores ao dado analisado, foram criadas redes bidirecionais recorrentes (BRNN, do inglês *Bidirectional Recurrent Neural Networks*). Nela, cada neurônio recebe tanto informações anteriores quanto posteriores da sequência que está sendo analisada.

2.4.4 Transformers

Similarmente às RNNs, os Transformers foram projetados com o objetivo de Processamento de Linguagem Natural (PLN). A diferença é que enquanto nas RNNs o processamento se dá de forma sequencial, os Transformers buscam dar mais prioridade ao contexto, podendo começar o processamento em qualquer parte do texto. Esta característica permite uma maior paralelização no processamento quando comparado às RNNs, reduzindo o tempo de computação e permitindo sua aplicação em bases de dados maiores [33].

Os Transformers funcionam baseado em um esquema *encoder-decoder*. Na primeira etapa do processamento do texto, as palavras são codificadas de tal forma que as partes consideradas importantes são ressaltadas. Isto é feito graça a um mecanismo de atenção que aprende quais os *tokens* que são mais relevantes para a predição. Após esta etapa, temos a etapa de decodificação. Novamente nesta etapa, os mecanismos de atenção são utilizados para extrair as informações mais importantes da entrada. Ao final, o modelo passa pelo processo de predição dos dados. Além disso, múltiplos *encoders* e *decoders* podem ser usados em um mesmo Transformer, o que apesar de tornar o processamento mais lento, aumenta sua performance significativamente.

De fato, os Transformers se consolidaram como o modelo mais importante da literatura para os problemas de PLN [35], com seu exemplo mais famoso sendo o BERT (do inglês *Bidirectional Encoder Representations from Transformers*) [8]. Como a etapa de treinamento exige um alto custo computacional, a prática mais adotada é o uso de modelos pré-treinados em bases de dados grandes, de preferência relacionadas ao problema estudado, onde é possível aprender no nível semântico da linguagem, por meio do treinamento *self-supervised*. Assim, o conjunto de treinamento de cada

problema específico é usado apenas para fazer alguns ajustes nas informações que já são conhecidas pelo modelo pré-treinado.

3 Conjunto de Dados

Neste trabalho, utilizamos a mesma base descrita em Kulmanov *et al.* [16], considerando apenas a ontologia de função molecular. Este artigo descreve a base derivada do desafio CAFA3 (do inglês *Critical Assessment of Functional Annotation*) [38], publicado em Setembro de 2016. Além disso, todas as funções que possuem menos de 50 proteínas na base foram removidas, restando 677 termos em diferentes níveis do DAG. A quantidade de proteínas usadas nos conjuntos de treinamento, validação e teste pode ser vista na Tabela 1, indicada como base original.

Tabela 1: Número de proteínas usadas na base original, tratada e profundidade 1.

Base	**Treinamento**	**Validação**	**Teste**	**Total**
Original	32.468	3.642	1.137	37.247
Tratada	32.421	3.587	1.137	37.145
Profundidade 1	29.014	3.219	1.036	33.269

Após a verificação das proteínas contidas nos conjuntos de treinamento, validação e teste da base original, notamos que existiam proteínas duplicadas em diferentes conjuntos, ou seja, existiam casos nos quais uma proteína estava tanto presente no conjunto de treinamento como no conjunto de validação e também proteínas que estavam tanto no conjunto de validação como no de teste. Isto é conceitualmente errado, uma vez que isto pode gerar o problema de *overfitting*. Portanto, para obtermos resultados mais fidedignos com a realidade do problema, optamos por também criar uma outra base - denominada *tratada* - na qual as proteínas duplicadas são removidas. O resultado final deste tratamento também pode ser visto na Tabela 1.

Na Figura 4, apresentamos um exemplo real de uma proteína presente na base de treinamento para ilustrar como cada proteína foi representada na base de dados. Nesta imagem, podemos ver que cada aminoácido é representado por uma letra, enquanto a proteína é uma sequência de aminoácidos. Cada função é indicada por um rótulo que foi atribuído pela ontologia genética. Indicamos com 0 caso a proteína não possua aquele rótulo e 1 caso contrário.

Sequência de aminoácidos	GO:0004497	GO:0003674	GO:0003824
...GGPGRSYTADAGYA...	0	1	1

Figura 4: Exemplo de uma proteína na base de dados com apenas 3 funções. Os valores 0 ou 1 indicam se a proteína possui ou não a função, respectivamente.

Além disso, fizemos algumas análises de dados na base tratada para entendermos melhor suas características. Na Figura 5 podemos observar o número de proteínas em cada agrupamento de número de rótulos, onde é possível notar que a maior parte das proteínas possuem de 1 a 10 rótulos. Outro ponto importante é que os conjuntos de treinamento, validação e teste visualmente possuem gráficos parecidos, o que é ponto positivo para que as predições sejam mais precisas.

Finalmente, como demonstrado na Seção 2, existem vários modelos que podem ser utilizados para predizer as funções das proteínas. Somado a isso, cada modelo pode possuir uma infinitude de diferentes valores de parâmetros. Como é computacionalmente inviável testar todos os modelos (e para cada modelo testar com vários parâmetros distintos), resolvemos simplificar a base tratada para buscarmos os melhores modelos com os melhores parâmetros.

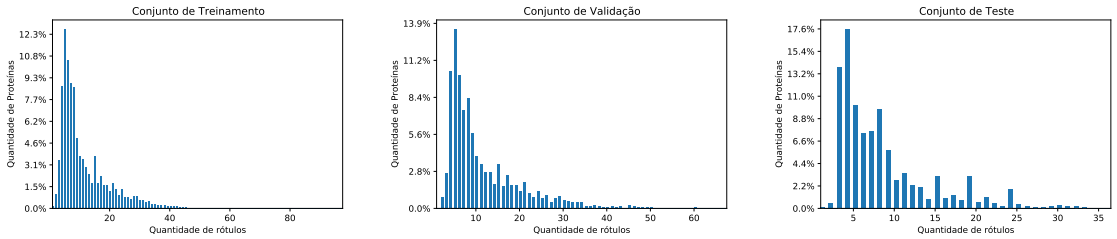

Figura 5: Relação entre quantidade de proteínas e quantidade de rótulos nos conjuntos de treinamento, validação e teste da base tratada.

Como explicado anteriormente, as funções estão organizadas de uma forma hierárquica, possuindo apenas um nó raiz. Sendo assim, escolhemos manter apenas as funções que são filhas do nó raiz, simplificando as 677 funções para apenas 17. Ao fazer esta simplificação, é possível que algumas proteínas não possuam nenhuma das 17 funções (proteínas que possuem apenas o nó raiz como termo assinalado na ontologia função molecular) e, portanto, também devemos remover estes casos. Além disso, como algumas proteínas possuem sequências de aminoácidos muito longas, optamos por filtrar proteínas com mais de 1.000 aminoácidos, o que ainda mantém cerca de 90% do total, conforme mostrado na Tabela 2. O resultado final é mostrado na Tabela 1 na base denominada *Profundidade 1*.

Tabela 2: Tamanho das sequências de aminoácidos por percentil na base tratada.

Percentil	**Tamanho da sequência**		
	Treinamento	**Validação**	**Teste**
5	130	138	104
10	178	183	120
30	317	321	225
50	433	429	344
70	590	578	509
90	1.019	1.010	920
100	34.350	35.213	8.891

4 Metodologia

Nesta seção, descrevemos o método proposto para a tarefa de predição de funções moleculares em proteínas, assim como a métrica de avaliação adotada e as ferramentas utilizadas neste trabalho.

4.1 Ferramentas Utilizadas

Os treinamentos e classificações foram feitos com a linguagem de programação Python 3.7. Para o treinamento dos modelos de aprendizado de máquina, utilizamos as bibliotecas `scikit-learn`[2] e `TensorFlow`[3], este último também sendo usado para os modelos de aprendizado profundo. Para

[2] `https://scikit-learn.org`
[3] `https://www.tensorflow.org`

os treinamento dos Transformers, utilizamos o `ktrain`[4] [21], enquanto os modelos pré-treinados podem ser encontrados no `Hugging Face`[5].

4.2 Tratamento da entrada

Vários modelos utilizados neste trabalho possuem duas restrições para a entrada: representação numérica e tamanho das proteínas. Como a entrada é uma sequência de aminoácidos no qual cada aminoácido é representado por uma letra, devemos realizar uma conversão para valores numéricos. Para isso, testamos duas abordagens: uma codificação numérica simples e a técnica *one-hot encoding*.

Na codificação numérica simples, transformamos cada letra em um número, como "A" em 1, "B" em 2 e assim por diante, seguindo a ordem alfabética. No *one-hot encoding*, cada letra que representa um aminoácido é convertida em um vetor binário de tamanho 26. Cada i-ésima posição do vetor representa a i-ésima letra do alfabeto. Assim, usamos 1 em uma posição i e 0 nas demais posições para indicar que o vetor representa a i-ésima letra.

Para a restrição de tamanho, a solução foi dividir sequências muito longas em trechos menores e atribuir para cada trecho os rótulos da proteína original. O tamanho máximo de cada trecho varia de acordo com o modelo aplicado. Além disso, alguns métodos exigem que todas as amostras possuam o mesmo tamanho. Neste caso, preenchemos as entradas vazias com 0.

4.3 Tratamento da saída

Como o tratamento da entrada adiciona novas amostras artificiais (os novos trechos das proteínas quebradas), também precisamos realizar um tratamento da saída para que as predições se refiram a entrada original. Para esta finalidade, analisamos e aplicamos 4 métodos diferentes de junção das predições de cada trecho para todos os rótulos: mediana, média, mínimo e máximo.

Após as predições dos modelos de aprendizado de máquina, precisamos realizar a etapa de atribuição de valores considerando a estrutura da ontologia genética. Portanto, devemos propagar a predição dos filhos para os pais, de modo que o valor do pai é o maior valor da predição de todos os filhos. Este procedimento é ilustrado na Figura 6.

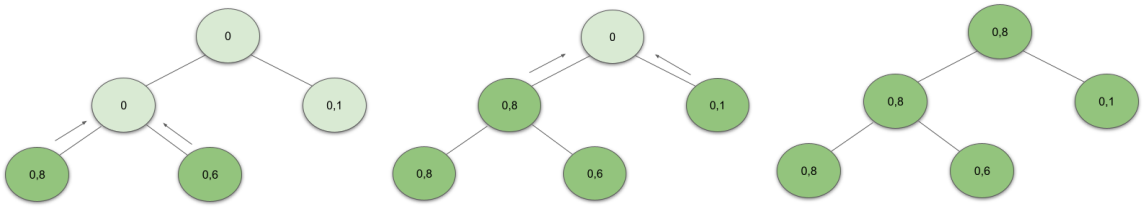

Figura 6: Procedimento de propagação das predições dos nós filhos para os nós pais. Em verde-escuro temos os nós que já foram processados enquanto os nós em verde-claro estão sendo processados.

4.4 *Baseline*

Para sermos capazes de concluir que os modelos estudados neste trabalho atingiram resultados satisfatórios, podemos comparar os valores obtidos com um algoritmo simples e ingênuo que realiza

[4] `https://github.com/amaiya/ktrain`
[5] `https://huggingface.co`

a mesma tarefa, que denominamos como *baseline*.

Neste trabalho, implementamos um algoritmo que prediz que uma proteína possui um rótulo baseado na frequência da presença deste rótulo no conjunto de treinamento. Assim, se um rótulo está presente em 10% das proteínas do conjunto de treinamento, o algoritmo prediz, para cada proteína no conjunto de validação e teste, a presença deste rótulo com 10% de chance.

4.5 Modelos para Classificação

Dividimos esta seção de acordo com as diferentes técnicas experimentadas. Assim temos: modelos de aprendizado de máquina, modelos de aprendizado profundo, Transformers e *ensemble* dos modelos de aprendizado de máquina.

Para ilustrar o procedimento padrão adotado pelos modelos, construímos o diagrama da Figura 7. Nela, podemos ver como uma sequência de aminoácidos, no exemplo com tamanho 8, passa desde o tratamento da entrada até o tratamento da saída para a predição de 2 funções. Neste caso, ilustrado na Figura 7, definimos o tamanho máximo de cada trecho como 3 e utilizamos a média como método de junção das predições de cada trecho.

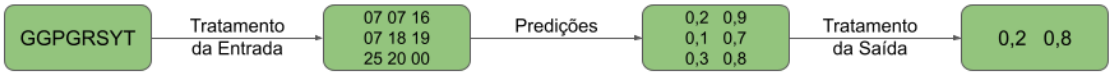

Figura 7: Metodologia aplicada para classificação das funções moleculares das proteínas.

4.5.1 Modelos de aprendizado de máquina

Esta subseção se refere aos métodos Random Forest, XGBoost e regressão logística. Nestes algoritmos, definimos 1.000 como o limite máximo de aminoácidos em cada trecho, pois, como explicado na Seção 3, cerca de 90% das proteínas possuem tamanho menor ou igual a 1.000 e, desta forma, poucas proteínas teriam que ser quebradas.

Para os 3 algoritmos, realizamos a busca dos hiper-parâmetros que otimizassem os resultados dos modelos. Em todos os casos, realizamos uma execução em paralelo com validação cruzada de 5 *folds*. No Random Forest, variamos entre 10, 50, 100, 200 e 500 os números de árvores utilizadas na predição. Já no XGBoost, o parâmetro verificado foi a taxa de aprendizado que variou entre os valores 0,001, 0,01, 0,1 e 1,0. Para a regressão logística, experimentamos tanto com os estimadores de penalidade L1 (regressão Lasso [28]) e L2 (regressão Ridge [12]) como também com o inverso da força de regularização com valores 0,01, 0,1, 1,0 e 10,0.

Uma última observação é que, como a abordagem de *one-hot encoding* aumenta consideravelmente a entrada do nosso problema, não foi possível aplicá-la para o caso da regressão logística. Para os outros dois métodos, testamos tanto o uso da abordagem de codificação simples como da técnica *one-hot encoding*.

4.5.2 Modelos de aprendizado profundo

Nesta subseção, explicamos como aplicamos os modelos Rede Inception Recorrente (IRN, do inglês *Inception Recurrent Network*) [7], BRNN e Inception-V4. Ao contrário dos algoritmos da seção anterior, estes modelos exigem que a entrada seja binária e, portanto, utilizamos apenas a técnica *one-hot encoding* no tratamento da entrada.

Nos experimentos do Inception-V4, variamos o número de camadas do tipo A, B e C e, para cada configuração, testamos a utilização da camada de *embedding*. Dois conjuntos de testes foram

organizados para a variação dos tipos de camadas. No primeiro, variamos de 1 a 5 (aumentando de 1 em 1) o número de camadas de um tipo (A, B ou C) mantendo os demais tipos com 0, para cada tipo. No segundo, mantemos o número de camadas de cada tipo igual e variamos de 1 a 5 (aumentando de 1 em 1).

No IRN, repetimos o processo descrito para o Inception-V4, mas, além disso, definimos o número de neurônios GRU como 100 e o número de camadas BRNN como 3.

Para o BRNN, variamos o número de camadas BRNN de 1 a 3 e, para cada número de camadas, variamos de 100 a 500 (aumentando de 100 em 100) o número de neurônios GRU. Além disso, para todas essas combinações, testamos usar ou não a camada de *embedding*.

Em todos os modelos utilizamos o tamanho de *batch* igual a 16 e a função de ativação para os blocos convolucionais foi a ReLU (do inglês *Rectified Linear Unit*). Já na camada de saída, a função de ativação foi a função sigmoide. Além disso, o modelo foi executado com o otimizador Adam [15], taxa de aprendizado de 0,001 e a função de custo entropia cruzada binária [24]. O treinamento foi realizado por 100 épocas com a técnica de parada precoce (do inglês *early stopping*) como critério de parada, realizando o encerramento deste processo caso o modelo não apresente melhorias após 10 épocas.

4.5.3 Transformers

No caso dos Transformers, 6 modelos pré-treinados foram utilizados para os testes: ProtBERT [9], ProtBERT-BFD [9], UncasedBERT [8], DistilUncasedBERT [27], XLM-RoBERTa [6] e RoBERTa [20].

Novamente, o tamanho de *batch* escolhido foi 16 e o treinamento foi realizado por 100 épocas com a técnica de parada precoce, realizando o encerramento deste processo caso o modelo não apresente melhorias após 1 época, com uma taxa de aprendizado de 0,00001. Além disso, por restrições de tempo de execução e também limite no tamanho máximo da sequência de aminoácidos, cada trecho das proteínas quebradas possuiu no máximo tamanho 100 e, por isso, também tivemos que aplicar os métodos de *ensemble* descritos na Subseção 4.3.

4.5.4 *Ensemble* dos métodos

Como última etapa, analisamos dois métodos de *ensemble* sobre as predições obtidas pelos Transformers pré-treinados com o ProtBert e ProtBERT-BFD.

Como temos as predições dadas pelos Transformers pré-treinados com ProtBert e com ProtBERT-BFD, o primeiro método consiste em unificar estas informações através das operações de mediana, média, mínimo e máximo considerando as predições de cada rótulo de cada trecho de uma mesma proteína. Denominamos este método como *ensemble* simples.

Além do *ensemble* simples, exploramos também a utilização de *embeddings* após o treinamento (ajuste fino) para a nossa base de dados. Mais especificamente, extraímos os *embeddings* de cada modelo através da função *tokenizer* que, resumidamente, consiste em 2 passos. No primeiro passo temos a *tokenization*, uma função que quebra a sequência de aminoácidos em *tokens*. No segundo passo, trocamos cada *token* por um valor, chamado *embedding*, que é a representação da sequência de aminoácidos em um vetor numérico de alta dimensão, que é utilizado na camada anterior da camada de saída.

É importante ressaltar que na *tokenization* também são adicionados dois *tokens* especiais: o CLS e o SEP. Como cada trecho possui tamanho 100, temos um vetor de *tokens* de tamanho 102 no qual na primeira posição temos o CLS, na última posição temos o SEP e nas demais temos a sequência de aminoácidos. A partir do token CLS, que é utilizado em tarefas de classificação,

conseguimos obter os valores de embedding de cada proteína. Este procedimento é ilustrado na Figura 8.

Figura 8: *Tokenizer* aplicado nas sequências de aminoácidos para a extração dos *embeddings*.

Após a extração dos *embeddings* das proteínas quebradas de cada modelo (ProtBert e ProtBERT-BFD), devemos realizar um *ensemble* nestes dados. Para isso, adotamos 2 abordagens, ilustradas pelas figuras 9 e 10. Na abordagem 1, primeiro fazemos o *ensemble* das proteínas quebradas através da média (obtendo assim os *embeddings* das proteínas originais de cada modelo), e depois aplicamos as operações de média, mínimo, máximo e concatenação entre os *embeddings* de cada modelo. Já na abordagem 2, aplicamos as operações de média, mínimo, máximo e concatenação entre os *embeddings* de cada modelo com as proteínas quebradas.

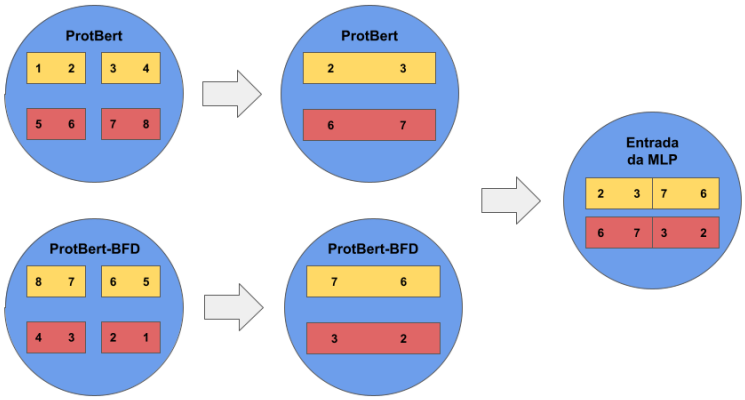

Figura 9: Exemplo de um *ensemble* dos *embeddings* gerados pelo ProtBert e pelo ProtBert-BFD com a abordagem 1 utilizando a concatenação. No caso ilustrado, temos 2 proteínas, uma em amarelo e outra em vermelho nas quais seus trechos quebrados são juntados através da média.

Finalmente, utilizamos os *embeddings* finais como entrada em uma MLP, que é o método responsável por realizar as predições. Neste método, variamos, para cada abordagem, o número de camadas de 1 até 3 (e aumentando de 1 em 1) com 1.000 neurônios em cada camada. Além disso, para a abordagem 2, como temos as predições das proteínas quebradas, usamos os métodos de *ensemble* descritos na Subseção 4.3.

4.6 Métrica de Avaliação

A métrica de avaliação utilizada neste trabalho foi o F_{max}, que é a mesma métrica utilizada no desafio CAFA3. Esta métrica utiliza os conceitos de precisão e revocação.

A precisão mede a quantidade de rótulos preditos corretamente pela quantidade rótulos preditos. Já a revocação mede a quantidade de rótulos preditos corretamente pela quantidade de rótulos que a proteína possui. A precisão e a revocação são definidas pelas equações 2 e 3, respectivamente.

$$pr(\tau) = \frac{1}{m(\tau)} \sum_{i=1}^{m(\tau)} \frac{\sum_f I(f \in P_i(\tau) \wedge f \in T_i)}{\sum_f I(f \in P_i(\tau))} \tag{2}$$

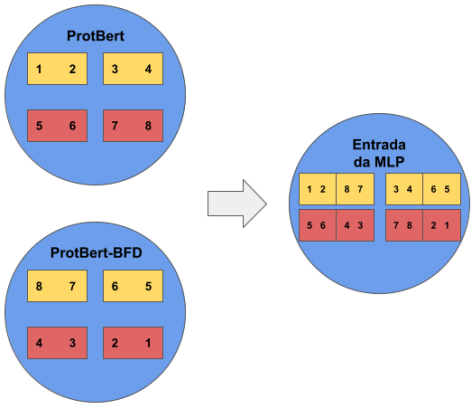

Figura 10: Exemplo de um *ensemble* dos *embeddings* gerados pelo ProtBert e pelo ProtBert-BFD com a abordagem 2 utilizando a concatenação. No caso ilustrado, temos 2 proteínas, uma em amarelo e outra em vermelho.

$$rc(\tau) = \frac{1}{n_e} \sum_{i=1}^{n} \frac{\sum_f I(f \in P_i(\tau) \wedge f \in T_i)}{\sum_f I(f \in T_i)} \tag{3}$$

Nas equações acima, $P_i(\tau)$ denota o conjunto dos termos que possuem predições maiores ou iguais a τ para a proteína i, T_i denota os rótulos que de fato aquela proteína possui, $m(\tau)$ é o número de sequências com pelo menos uma predição maior ou igual a τ, $I(\cdot)$ é uma função identidade, n_e é o número de proteínas usadas na avaliação usados em um modo de avaliação e f é um rótulo da ontologia.

Com estas definições, podemos descrever a métrica de avaliação do CAFA3 como sendo a média harmônica entre a precisão e a revocação considerando diversos valores do limiar τ, representado na Equação 4. Neste trabalho variamos o limiar $\tau \in [0, 1]$ com passo de 0,01.

$$F_{max} = max_\tau \{ \frac{2 \times pr(\tau) \times rc(\tau)}{pr(\tau) + rc(\tau)} \} \tag{4}$$

5 Resultados e Discussão

Nesta seção, apresentamos e discutimos os resultados dos métodos propostos no conjunto com profundidade 1, na base tratada e na base original.

5.1 Profundidade 1

Inicialmente, avaliamos os classificadores no conjunto de dados com profundidade 1 da ontologia de função molecular. Na Tabela 3, apresentamos os resultados dos nossos classificadores, na qual podemos notar que os algoritmos que utilizam aprendizado profundo obtiveram os melhores resultados.

Dentre os algoritmos de aprendizado profundo, o classificador do tipo Transformers obteve o melhor resultado. Mais especificamente neste método, o modelo pré-treinado que conseguiu atingir o valor de 0,890 foi o ProtBERT, conforme mostra a Tabela 4. Além disso, é importante compararmos este valor com o obtido pelo método *baseline* para sermos capazes de afirmar que o modelo apresenta uma melhora significativa. Como o método *baseline* obteve um F_{max} de 0,789, temos um aumento

expressivo de 0,101 e, portanto, o suficiente para concluirmos que os classificadores possuem uma diferença significativa.

Tabela 3: F_{max} obtido por cada modelo para a profundidade 1.

Modelo	F_{max}
Baseline	0,789
Aprendizado de Máquina	
Random Forest	0,815
XGBoost	0,800
Regressão Logística	0,786
Aprendizado Profundo	
Transformers	0,890
IRN	0,872
BRNN	0,868
Inception-V4	0,864

Entretanto, devemos ressaltar que estas predições se referem a apenas os filhos do nó raiz. Além de estarmos tratando apenas 17 termos (ao invés do total de 677), como os filhos do nó raiz se referem a termos mais genéricos, eles estão mais presentes nas proteínas do que termos mais profundos na DAG. Portanto, como temos mais amostras no nosso conjunto de treinamento para cada termo, as predições tendem a ser mais precisas em relação ao DAG completo.

Assim, o objetivo principal desta primeira parte do trabalho foi inferir quais eram os parâmetros que geravam os melhores resultados para cada método. Como estamos tratando de apenas 17 rótulos, os tempos de execução eram relativamente pequenos, o que nos permitiu testar uma boa quantidade de parametrizações diferentes para cada modelo.

Para os classificadores de aprendizado de máquina (Random Forest, XGBoost e Regressão Logística), a representação que obteve os melhores resultados foi a codificação simples dos aminoácidos através de um dicionário. Uma provável explicação para este fato é que ao utilizarmos *one-hot encoding* estamos aumentando o tamanho da entrada na ordem de milhões, além de que a maioria das caraterísticas possuem valores iguais a 0. Somado a estes fatos, também temos que quebrar a proteína em sequências menores e usar a técnica da janela deslizante, o que torna o processo de treinamento lento e impreciso.

Ainda sim, é importante notar que a codificação simples também possui uma desvantagem. Como os números possuem uma ordem sequencial, o algoritmo tende a assumir que o aminoácido "A" possui uma característica mais próxima do aminoácido "B" do que "Z", o que na prática não é necessariamente verdade. Acreditamos que essa pode ser uma das razões na qual os modelos de aprendizado de máquina obtiveram os piores resultados (a Regressão Logística, inclusive, pior que o *baseline*).

Considerando os modelos de aprendizado profundo, o classificador IRN atingiu os melhores resultados com 3 camadas do tipo "B" e com 100 neurônios nas camadas bidirecionais recorrentes, o classificador BRNN obteve os melhores resultados com 2 camadas com 400 neurônios cada, e o Inception-V4 atingiu os melhores resultados com 5 camadas do tipo "B". Assim, vemos uma clara vantagem no uso de camada do tipo "B" para o problema estudado neste trabalho.

Em relação aos Transformers, a Tabela 4 apresenta o F_{max} para as diferentes arquiteturas pré-treinadas. Nesta tabela, podemos ver que os modelos ProtBERT e ProtBERT-BFD se destacam dos demais. Estes modelos foram pré-treinados em bases com sequências de aminoácidos, o que é mais adequado ao nosso problema, e, por isso, os seus resultados obtidos atingiram melhores valores

quando comparado aos demais, que foram pré-treinados em bases de dados com textos em inglês. Além disso, ressaltamos que para todos os modelos, o melhor método de *ensemble* das predições das proteínas quebradas foi a através da média.

Tabela 4: F_{max} obtido por cada modelo pré-treinado para a profundidade 1.

Modelo pré-treinado	F_{max}
ProtBERT [9]	0,890
ProtBERT-BFD [9]	0,887
UncasedBERT [8]	0,859
DistilUncasedBERT [27]	0,855
XLM-RoBERTa [6]	0,852
RoBERTa [20]	0,848

5.2 Tratada

Como observado na seção anterior, os modelos de aprendizado profundo são claramente superiores aos modelos de aprendizado de máquina para o problema de predição de funções moleculares em proteínas. Portanto, optamos por aplicar apenas os modelos de aprendizado profundo na base tratada, utilizando as melhores parametrizações encontradas para a profundidade 1. Como os modelos baseados em Transformers obtiveram os melhores resultados, aplicamos os dois melhores modelos, que foram pré-treinados em base de proteínas (ProtBERT e ProtBERT-BFD), e o melhor modelo treinado em língua inglesa (UncasedBERT). Estes resultados são apresentados na Tabela 5.

Tabela 5: F_{max} obtido pelos modelos de aprendizado profundo para a base tratada.

Modelo	F_{max}
Baseline	0,417
Aprendizado Profundo	
ProtBERT-BFD [9]	0,620
ProtBERT [9]	0,609
UncasedBERT [8]	0,515
IRN [7]	0,562
Inception-V4 [30]	0,524

Uma última ressalva pode ser feita para o modelo BRNN. Embora ele tenha atingido um bom F_{max} de 0,868 na profundidade 1, não foi possível executar o mesmo modelo para o DAG completo, pois o modelo não apresentava uma melhora no F_{max} desde a primeira época, ou seja, atingiu (provavelmente) um ótimo local. Uma hipótese que pode explicar este fato é um problema clássico de redes neurais recorrentes: o erro do gradiente se extingue de maneira muita rápida. Por esta razão, o BRNN não está presente em nenhuma das tabelas desta seção.

Na sequência, realizamos o *ensemble* entre os dois melhores modelos, ProtBERT e ProtBERT-BFD. Os resultados da fusão são apresentados na Tabela 6, considerando tanto a fusão pelo método de *ensemble* simples quanto utilizando uma rede neural do tipo MLP. Como discutido na Seção 4, para cada método de junção entre os embeddings do ProtBERT e do ProtBERT-BFD (concatenação, média, mínimo e máximo) utilizamos duas abordagens, a primeira (indicada como abordagem 1) considera o *embedding* da proteína completa predito por cada modelo, após a junção

dos trechos, e a segunda (indicada como abordagem 2) utiliza o *embedding* de cada trecho (mesmo trecho para cada modelo) e depois juntado para ter a predição por proteína. Em relação a junção dos pedaços de cada proteína, resolvemos adotar a média entre os trechos, já que esta abordagem obteve melhores resultados na profundidade 1. Utilizamos este método tanto para a abordagem 1 quanto para a abordagem 2.

Tabela 6: F_{max} obtido pelo *ensemble* simples e pelo MLP para a base tratada.

Modelo	Abordagem	Nº camadas	F_{max}
Ensemble simples	—	—	0,620
MLP			
Média	1	1	0,650
Concatenação	1	2	0,649
Mínimo	1	1	0,648
Máximo	1	2	0,647
Concatenação	2	1	0,534
Média	2	2	0,553
Máximo	2	2	0,520
Mínimo	2	3	0,519

Pelos resultados, podemos observar que, embora tenhamos testado o MLP com 1, 2 e 3 camadas, apenas uma configuração com 3 camadas obteve o melhor resultado, o que justifica a razão por não termos testados com mais camadas. Ademais, todos os testes feitos com a abordagem 1 foram superiores a abordagem 2. Este fato nos sugere que ao quebrarmos a proteína em vários trechos (e depois predizer os rótulos de cada trecho) estamos perdendo uma informação importante no treinamento: os vizinhos da proteína original serem separados em diferentes amostras.

5.3 Aplicando o Melhor Modelo no Conjunto de Teste

Por último, aplicamos o melhor modelo, o MLP com 1 camada que utiliza como entrada a média dos *embeddings* do ProtBERT e ProtBERT-BFD para cada proteína, ou seja, treinando o MLP com o *embedding* da proteína como um todo, na base de teste, e comparamos com o *baseline*. Os resultados são demonstrados na Tabela 7.

Tabela 7: F_{max} obtido pelo baseline e pelo nosso método para a base tratada no conjunto de teste.

Modelo	F_{max}
Nosso método	0,562
Baseline	0,446

Com estes resultados finais, observamos um aumento de 0,116 no F_{max} obtido pelo nosso modelo em relação ao *baseline*. Assim, concluímos que a evolução observada na profundidade 1 (de 0,101) não apenas se manteve como aumentou, consolidando assim o modelo apresentado neste trabalho como um método viável para uso em predições das funções moleculares das proteínas.

Além disso, para ilustrarmos melhor o comportamento deste modelo quando aplicado no conjunto de teste, apresentamos a Figura 11. Neste gráfico, podemos observar que a revocação exibe uma queda suave na medida em que o limiar das predições aumenta, o que indica que as predições realizadas pelo modelo possuem um alto grau de confiança, pois o número de rótulos que não são

mais considerados presentes, na medida em que o limiar aumenta, é baixo. A precisão, por sua vez, segue uma trajetória ao contrário da revocação, exibindo um crescimento mais acentuado, o que é esperado dado que apenas as predições com alta confiança são consideradas. No caso do F, que é uma média harmônica das duas medidas anteriores, naturalmente segue o padrão do menor valor, a revocação. Na Figura 11, o limiar que obteve o maior valor de F foi 0,21.

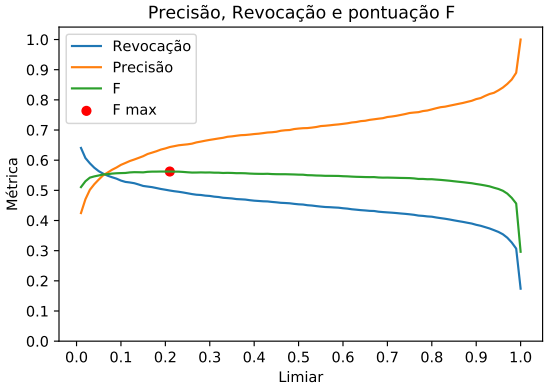

Figura 11: Precisão, revocação, F e F_{max} do nosso modelo no conjunto de teste da base tratada.

5.4 Comparação com DeepGOPlus

Para compararmos o nosso classificador com o DeepGOPlus, repetimos o procedimento realizado para a base tratada, mas, desta vez, utilizando a base original. O resultado obtido e sua comparação com o resultado do DeepGOPlus é exibido na Tabela 8. Com um aumento de 0,060 no F_{max}, podemos concluir que o modelo proposto neste trabalho é superior ao do DeepGOPlus (que por sua vez, é o melhor resultado na literatura para esta base de dados).

Tabela 8: F_{max} obtido pelo DeepGOPlus e pelo nosso modelo no conjunto de teste da base original.

Modelo	F_{max}
Nosso modelo	0,617
DeepGOPlus [16]	0,557
Baseline	0,446

Uma última observação pode ser feita em relação a diferença dos resultados alcançados na base original e na base tratada. Como a base original possui proteínas duplicadas nos conjuntos de treinamento e validação, treinamento e teste, e validação e teste, é de se esperar que as predições realizadas na base original sejam mais precisas, pois o modelo foi otimizado para as proteínas do teste. Como o nosso modelo atingiu um F_{max} de 0,562 na base tratada e 0,616 na base original, temos que esta hipótese foi confirmada empiricamente. Este fato evidencia que o resultado obtido pelo DeepGOPlus foi influenciado pelas proteínas duplicadas e, portanto, em uma base devidamente tratada seu F_{max} provavelmente seria ainda menor.

Repetindo o processo realizado na seção anterior, apresentamos na Figura 12 o gráfico da precisão, revocação e F_{max} do nosso modelo no conjunto de teste da base original. Comparando esta imagem com a a Figura 11, observamos que a principal diferença se encontra nos limiares baixos. Enquanto na base original quase todos os rótulos são preditos como presentes para limiares

baixos (por causa da precisão baixa e revocação alta), o mesmo não é verdade para a base tratada. Novamente, este fato indica que as predições realizadas na base tratada ou são valores muito próximos de 0 ou são valores muito próximos de 1. Esta característica é comprovada na Tabela 9. Na Figura 12, o limiar que obteve o maior valor de F foi 0,44.

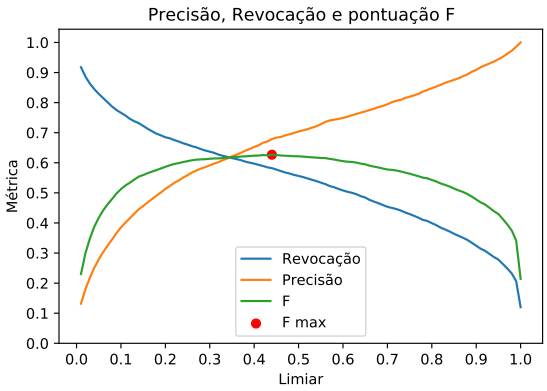

Figura 12: Precisão, revocação, F e F_{max} do nosso modelo no conjunto de teste da base original.

Tabela 9: Número de valores de predições do nosso modelo no conjunto de teste da base tratada nos intervalos [0; 0,2), [0,2; 0.8] e (0,8; 1,0].

Predições $<$ 0,2	**Predições $\geq 0,2$ e $\leq 0,8$**	**Predições $>$ 0,8**
98.99%	0.35%	0.66%

6 Conclusões e Trabalhos Futuros

O problema de predição das funções moleculares em proteínas é um dos grandes desafios atuais na área de biologia. Na data de publicação deste trabalho, o número de proteínas que ainda não foram rotuladas com a ontologia genética é superior a 200 milhões. Como os testes em laboratório que determinam empiricamente as funções moleculares de proteínas são lentos e, geralmente, custosos, um novo método que realize este procedimento com alta precisão e de forma automática, seria grande valia.

Neste trabalho, apresentamos e discutimos um novo modelo de aprendizado de máquina que usa os *embeddings* gerados pôr dois Transformers, um pré-treinado com o ProtBERT e outro com ProtBERT-BFD, como entrada para uma MLP. Com nosso método, fomos capazes de atingir um F_{max} de 0,562. Além disso, aplicando nosso modelo na base utilizada pelo DeepGOPlus, obtemos um F_{max} de 0,617, superando o valor obtido pelo DeepGOPlus de 0,557 que, até então, era o estado da arte.

Como trabalhos futuros, sugerimos a investigação da quantidade de número de neurônios usados na MLP, assim como a utilização de outros métodos de aprendizado de máquina para a realização da predição do *ensemble* dos *embeddings* gerados pelos Transformers.

Referências

[1] Stephen F. Altschul, Thomas L. Madden, Alejandro A. Schäffer, Jinghui Zhang, Zheng Zhang, Webb Miller, and David J. Lipman. Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Research*, 25(17):3389–3402, September 1997.

[2] Christian B. Anfinsen. Principles that Govern the Folding of Protein Chains. *Science*, 181(4096):223–230, July 1973.

[3] Michael Ashburner, Catherine A. Ball, Judith A. Blake, David Botstein, Heather Butler, J. Michael Cherry, Allan P. Davis, Kara Dolinski, Selina S. Dwight, Janan T. Eppig, Midori A. Harris, David P. Hill, Laurie Issel-Tarver, Andrew Kasarskis, Suzanna Lewis, John C. Matese, Joel E. Richardson, Martin Ringwald, Gerald M. Rubin, and Gavin Sherlock. Gene Ontology: tool for the unification of biology. *Nature Genetics*, 25(1):25–29, May 2000.

[4] Leo Breiman. Arcing the edge. Technical report, Technical Report 486, Statistics Department, University of California at Berkeley, 1997.

[5] Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. *arXiv:1406.1078*, September 2014.

[6] Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. Unsupervised Cross-lingual Representation Learning at Scale. *arXiv:1911.02116*, April 2020.

[7] Gabriel Bianchin de Oliveira, Helio Pedrini, and Zanoni Dias. Ensemble of Template-Free and Template-Based Classifiers for Protein Secondary Structure Prediction. *International Journal of Molecular Sciences (IJMS)*, 22(21):11449, October 2021.

[8] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *arXiv:1810.04805*, May 2019.

[9] Ahmed Elnaggar, Michael Heinzinger, Christian Dallago, Ghalia Rihawi, Yu Wang, Llion Jones, Tom Gibbs, Tamas Feher, Christoph Angerer, Martin Steinegger, Debsindhu Bhowmik, and Burkhard Rost. ProtTrans: Towards Cracking the Language of Life's Code Through Self-Supervised Deep Learning and High Performance Computing. *arXiv:2007.06225*, May 2021.

[10] Evelyn Fix and J. L. Hodges. Discriminatory Analysis. Nonparametric Discrimination: Consistency Properties. *International Statistical Review / Revue Internationale de Statistique*, 57(3):238, December 1989.

[11] Areski Flissi, Emma Ricart, Clémentine Campart, Mickael Chevalier, Yoann Dufresne, Juraj Michalik, Philippe Jacques, Christophe Flahaut, Frederique Lisacek, Valérie Leclère, et al. Norine: update of the nonribosomal peptide resource. *Nucleic Acids Research*, 48(D1):D465–D469, 2020.

[12] Marvin H. J. Gruber. *Improving efficiency by shrinkage: the James-Stein and ridge regression estimators*. Number v. 156 in Statistics, textbooks and monographs. Marcel Dekker, New York, 1998.

[13] Joseph M. Hilbe. *Logistic regression models*. Chapman and Hall/CRC, 2009.

[14] Sepp Hochreiter and Jürgen Schmidhuber. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780, November 1997.

[15] Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. *arXiv:1412.6980*, January 2017.

[16] Maxat Kulmanov and Robert Hoehndorf. DeepGOPlus: improved protein function prediction from sequence. *Bioinformatics*, 36(2):422–429, July 2019.

[17] Maxat Kulmanov, Mohammed Asif Khan, and Robert Hoehndorf. DeepGO: predicting protein functions from sequence and interactions using a deep ontology-aware classifier. *Bioinformatics*, 34(4):660–668, February 2018.

[18] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *IEEE*, 86(11):2278–2324, November 1998.

[19] Albert L. Lehninger, David L. Nelson, and Michael M. Cox. *Lehninger principles of biochemistry*. W.H. Freeman, New York, 4th ed edition, 2005.

[20] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. RoBERTa: A Robustly Optimized BERT Pretraining Approach. *arXiv:1907.11692*, July 2019.

[21] Arun S. Maiya. ktrain: A low-code library for augmented machine learning. *arXiv:2004.10703*, 2020.

[22] Peter McCullagh and John A. Nelder. *Generalized linear models*. Number 37 in Monographs on statistics and applied probability. Chapman & Hall/CRC, Boca Raton, 2nd ed edition, 1998.

[23] Tom M. Mitchell. *Machine Learning*. McGraw-Hill series in computer science. McGraw-Hill, New York, 1997.

[24] Kevin P. Murphy. *Machine learning: a probabilistic perspective*. Adaptive computation and machine learning series. MIT Press, Cambridge, MA, 2012.

[25] Lior Rokach and Oded Maimon. *Data mining with decision trees: theroy and applications*. Number v. 69 in Series in machine perception and artificial intelligence. World Scientific, Singapore; Hackensack, NJ, 2008.

[26] Frank Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6):386, 1958.

[27] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. *arXiv:1910.01108*, February 2020.

[28] Fadil Santosa and William W. Symes. Linear Inversion of Band-Limited Reflection Seismograms. *SIAM Journal on Scientific and Statistical Computing*, 7(4):1307–1330, October 1986.

[29] Nils Strodthoff, Patrick Wagner, Markus Wenzel, and Wojciech Samek. UDSMProt: universal deep sequence models for protein classification. *Bioinformatics*, 36(8):2401–2409, April 2020.

[30] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alex Alemi. Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning. *arXiv:1602.07261*, August 2016.

[31] Tin Kam Ho. Random decision forests. In *3rd International Conference on Document Analysis and Recognition (ICDAR)*, volume 1, pages 278–282, Montreal, Que., Canada, 1995. IEEE Comput. Soc. Press.

[32] UniProt. UniProt Database. `https://www.uniprot.org/`, 2021. Online; acessado em 03 de Dezembro de 2021.

[33] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems (NIPS)*, pages 5998–6008, 2017.

[34] Robert Franklin Weaver. *Molecular Biology*. McGraw-Hill, New York, 5th edition, 2012.

[35] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. Transformers: State-of-the-Art Natural Language Processing. In *Conference on Empirical Methods in Natural Language Processing (EMNLP): System Demonstrations*, pages 38–45, Online, 2020. Association for Computational Linguistics.

[36] Ronghui You, Xiaodi Huang, and Shanfeng Zhu. DeepText2GO: Improving large-scale protein function prediction with deep semantic text representation. *Methods*, 145(1):82–90, August 2018.

[37] Ronghui You, Zihan Zhang, Yi Xiong, Fengzhu Sun, Hiroshi Mamitsuka, and Shanfeng Zhu. GOLabeler: improving sequence-based large-scale protein function prediction by learning to rank. *Bioinformatics*, 34(14):2465–2473, July 2018.

[38] Naihui Zhou, Yuxiang Jiang, Timothy R. Bergquist, Alexandra J. Lee, Balint Z. Kacsoh, Alex W. Crocker, Kimberley A. Lewis, George Georghiou, Huy N. Nguyen, Md Nafiz Hamid, Larry Davis, Tunca Dogan, Volkan Atalay, Ahmet S. Rifaioglu, Alperen Dalkıran, Rengul Cetin Atalay, Chengxin Zhang, Rebecca L. Hurto, Peter L. Freddolino, Yang Zhang, Prajwal Bhat, Fran Supek, José M. Fernández, Branislava Gemovic, Vladimir R. Perovic, Radoslav S. Davidović, Neven Sumonja, Nevena Veljkovic, Ehsaneddin Asgari, Mohammad R.K. Mofrad, Giuseppe Profiti, Castrense Savojardo, Pier Luigi Martelli, Rita Casadio, Florian Boecker, Heiko Schoof, Indika Kahanda, Natalie Thurlby, Alice C. McHardy, Alexandre Renaux, Rabie Saidi, Julian Gough, Alex A. Freitas, Magdalena Antczak, Fabio Fabris, Mark N. Wass, Jie Hou, Jianlin Cheng, Zheng Wang, Alfonso E. Romero, Alberto Paccanaro, Haixuan Yang, Tatyana Goldberg, Chenguang Zhao, Liisa Holm, Petri Törönen, Alan J. Medlar, Elaine Zosa, Itamar Borukhov, Ilya Novikov, Angela Wilkins, Olivier Lichtarge, Po-Han Chi, Wei-Cheng Tseng, Michal Linial, Peter W. Rose, Christophe Dessimoz, Vedrana Vidulin, Saso Dzeroski, Ian Sillitoe, Sayoni Das, Jonathan Gill Lees, David T. Jones, Cen Wan, Domenico Cozzetto, Rui Fa, Mateo Torres, Alex Warwick Vesztrocy, Jose Manuel Rodriguez, Michael L. Tress, Marco Frasca, Marco Notaro, Giuliano Grossi, Alessandro Petrini, Matteo Re, Giorgio Valentini, Marco Mesiti, Daniel B. Roche, Jonas Reeb, David W. Ritchie, Sabeur Aridhi, Seyed Ziaeddin Alborzi, Marie-Dominique Devignes, Da Chen Emily Koo, Richard Bonneau, Vladimir Gligorijević, Meet Barot, Hai Fang, Stefano Toppo, Enrico Lavezzo, Marco Falda, Michele

Berselli, Silvio C.E. Tosatto, Marco Carraro, Damiano Piovesan, Hafeez Ur Rehman, Qizhong Mao, Shanshan Zhang, Slobodan Vucetic, Gage S. Black, Dane Jo, Erica Suh, Jonathan B. Dayton, Dallas J. Larsen, Ashton R. Omdahl, Liam J. McGuffin, Danielle A. Brackenridge, Patricia C. Babbitt, Jeffrey M. Yunes, Paolo Fontana, Feng Zhang, Shanfeng Zhu, Ronghui You, Zihan Zhang, Suyang Dai, Shuwei Yao, Weidong Tian, Renzhi Cao, Caleb Chandler, Miguel Amezola, Devon Johnson, Jia-Ming Chang, Wen-Hung Liao, Yi-Wei Liu, Stefano Pascarelli, Yotam Frank, Robert Hoehndorf, Maxat Kulmanov, Imane Boudellioua, Gianfranco Politano, Stefano Di Carlo, Alfredo Benso, Kai Hakala, Filip Ginter, Farrokh Mehryary, Suwisa Kaewphan, Jari Björne, Hans Moen, Martti E.E. Tolvanen, Tapio Salakoski, Daisuke Kihara, Aashish Jain, Tomislav Šmuc, Adrian Altenhoff, Asa Ben-Hur, Burkhard Rost, Steven E. Brenner, Christine A. Orengo, Constance J. Jeffery, Giovanni Bosco, Deborah A. Hogan, Maria J. Martin, Claire O'Donovan, Sean D. Mooney, Casey S. Greene, Predrag Radivojac, and Iddo Friedberg. The CAFA challenge reports improved protein function prediction and new functional annotations for hundreds of genes through experimental screens. *Genome Biology*, 20(1):244, December 2019.