UNIVERSIDADE ESTADUAL DE MARINGÁ
PROGRAMA DE INICIAÇÃO CIENTÍFICA - PIC
DEPARTAMENTO DE INFORMÁTICA
ORIENTADOR(A): Profa. Dra. Josiane Melchiori Pinheiro
CO-ORIENTADOR(A): João Alencar Pamphile
ACADÊMICO(S): Gustavo Henrique Ferreira Cruz
Vinícius Menossi

Um software para identificação de regiões codantes em genes de fungos filamentosos e sua tradução para os polipeptídeos correspondentes

Maringá, 30 de Agosto de 2021

UNIVERSIDADE ESTADUAL DE MARINGÁ
PROGRAMA DE INICIAÇÃO CIENTÍFICA - PIC
DEPARTAMENTO DE INFORMÁTICA
ORIENTADOR(A): Profa. Dra. Josiane Melchiori Pinheiro
CO-ORIENTADOR(A): João Alencar Pamphile
ACADÊMICO(S): Gustavo Henrique Ferreira Cruz
Vinícius Menossi

Um software para identificação de regiões codantes em genes de fungos filamentosos e sua tradução para os polipeptídeos correspondentes

Relatório contendo os resultados finais do projeto de iniciação científica vinculado ao programa PIC-UEM.

Maringá, 30 de Agosto de 2021

Resumo

Este trabalho tem por objetivo identificar as regiões de *splicing* em sequências de RNA, usando uma abordagem baseada no aprendizado de máquina supervisionado, mais especificamente, no modelo chamado *Conditional Random Fields* (CRF). A pesquisa foi realizada tendo como base o trabalho de Ferreira (2019), o qual também fez uso do CRF para identificar as regiões codantes e não codantes (e consequentemente as regiões de *splicing* na sequência). Com o intuito de avançar na solução do problema e melhorar os resultados obtidos por Ferreira (2019), este trabalho alterou a abordagem do problema, criando uma nova base de treinamento para o CRF e tornando a implementação da solução mais legível e modular. Os resultados obtidos pelos modelos treinados na identificação das regiões de *splicing* foram bastante satisfatórios para o fungo *Colletotrichum* na proteína actina e menos promissores nas regiões codantes do fungo *Diaporthe* na proteína beta tubulina.
Palavras-chave: RNA, Regiões codantes, Aprendizado de máquina, Regiões de *splicing*, Conditional Random Fields, fungos filamentosos.

Sumário

1. Introdução

Identificar as regiões codantes e não codantes do RNA - também conhecida como as regiões de *splicing*, é uma tarefa que, atualmente, é pouco automatizada. Um pesquisador da área da biologia, biotecnologia e afins, hoje, tem que realizar todo o processo de análise e identificação destas regiões através do uso metódico e manual de técnicas da área, como, por exemplo, a identificação por meio da análise da proteína resultante.

Além disso, a identificação das regiões de *splicing* ainda é um tema pouco explorado no campo científico. Alguns trabalhos como o de Ferreira (2019) e Ozyilmaz & Palavaroglu (2008) apontam que esta é uma área de pesquisa com bastante potencial para a aplicação de técnicas computacionais, como o Aprendizado de Máquina (AM). Os estudos feitos apontam para padrões nas regiões codantes, mais especificamente, na identificação dos íntrons. Modelos de redes neurais e baseados na cadeia oculta de Markov (como o algoritmo *Conditional Random Fields*) conseguiram resultados significativos na identificação desses padrões.

O presente trabalho busca continuar os estudos iniciados por Ferreira (2019), na identificação das regiões codantes no RNA. Na abordagem feita por Ferreira (2019), os dados coletados do GenBank eram usados para treinar um modelo, capaz de identificar estas regiões e retornar dois tipos de respostas: “intron” e “not-intron”, porém, a abordagem escolhida resultava em um modelo que contava com uma quantidade muito maior de exemplos da classe “not-intron”, o que fez com que o resultado não fosse satisfatório.

Neste trabalho, uma nova base de dados foi criada, identificando três tipos de classes: íntron, éxon e *neither*. Além disso, o código de preparação da base foi reorganizado em módulo para tornar o software mais manutenível e servir como base para os trabalhos futuros do Grupo de Estudos no qual este trabalho se insere.

Este relatório está organizado da seguinte forma: na seção 2 são retomados os objetivos do projeto, na seção 3 é apresentada a fundamentação teórica necessária para o entendimento do trabalho, na seção 4 são apresentados os módulos e algoritmos desenvolvidos e, por fim, na seção 5 são apresentados os resultados obtidos.

2. Objetivos

Este trabalho tem por objetivo geral continuar o trabalho iniciado por Ferreira (2019) na identificação de íntrons e éxons em genes de fungos filamentosos. Como objetivos específicos pretende-se:

- Construir um módulo de tradução das regiões codantes do gene (éxons) para as proteínas correspondentes, podendo, com base na proteína obtida, validar os resultados obtidos por Ferreira (2019) para a identificação dos íntrons;
- Investigar novas formas de estruturação da base de dados de entrada do algoritmo *Conditional Random Fields*, procurando melhorar seu desempenho e eficiência na identificação dos íntrons e éxons;

A principal motivação desta pesquisa é a dificuldade que pesquisadores, estudiosos, alunos e comunidade científica no geral ainda têm ao ter que descobrir possíveis áreas de *splicing* do RNA, ação esta que ainda é realizada manualmente. O modelo proposto e estudado neste projeto tenta se adaptar, aprender e prever estas regiões, facilitando assim a identificação das mesmas. Além disso, com a proposta de implementação deste trabalho, tem-se a possibilidade de expandir os estudos para diferentes espécies filamentosas ou proteicas.

3. Fundamentação Teórica

Nesta seção serão apresentados os conceitos necessários para o entendimento do trabalho desenvolvido, bem como a posterior avaliação dos resultados obtidos.

Sendo aplicado em âmbito da biologia e biotecnologia, este trabalho se voltou para a necessidade de conceitos relacionados a estas áreas, além da fundamentação teórica da área de Aprendizado de Máquina e do desenvolvimento e realização do próprio projeto e seus módulos.

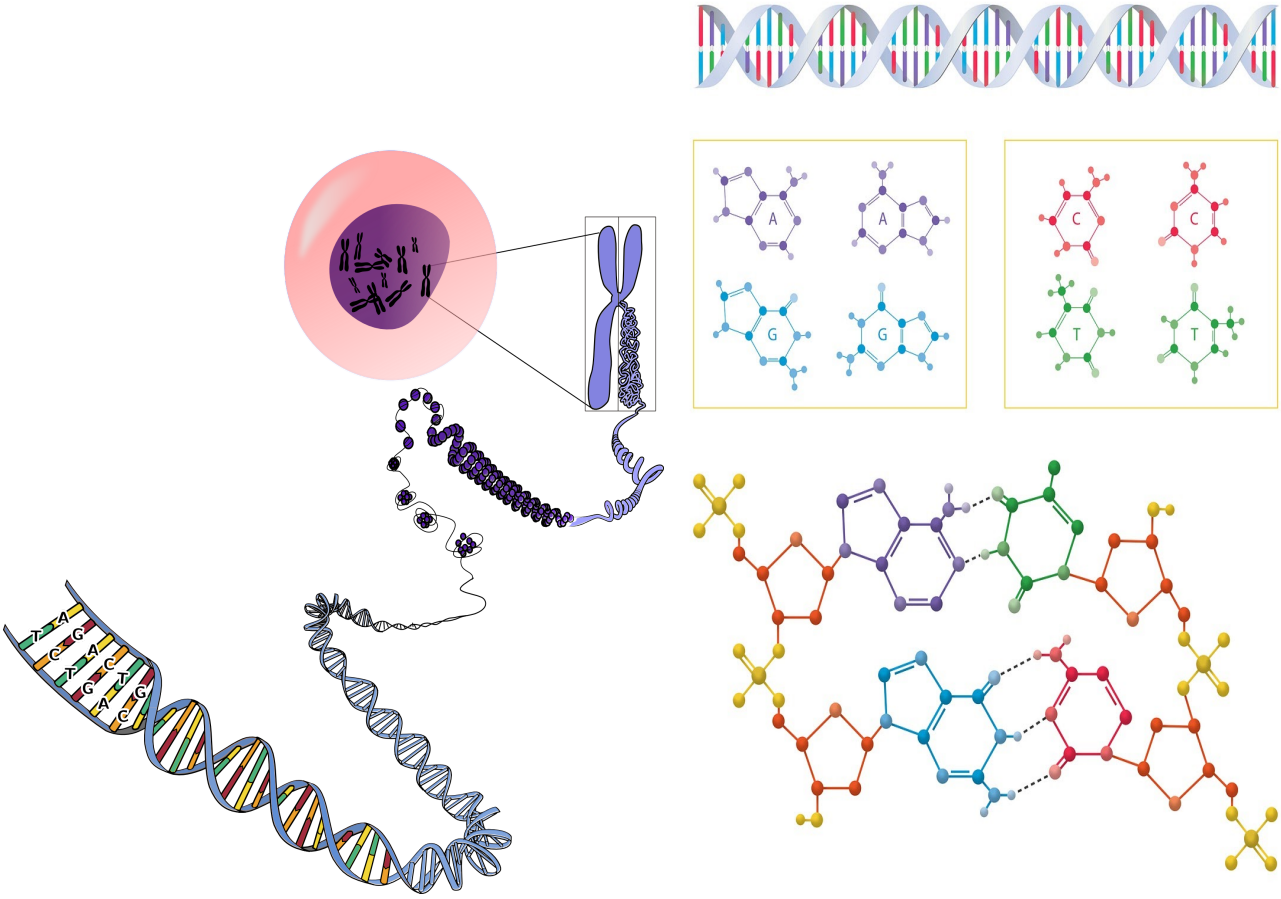
3.1. Conceitos da biologia

Para o desenvolvimento e implementação foi necessário compreender alguns conceitos na geração de proteínas, sendo alguns deles, o conceito de gene, DNA, RNA, nucleotídeos, processos de transcrição, e suas etapas, como o *splicing* e a tradução.

3.1.1. DNA

O ácido desoxirribonucleico ou em inglês “*deoxyribonucleic acid*” (DNA), descoberto originalmente por Rosalind Franklin (1920-1958), está localizado principalmente nos cromossomos e mitocôndrias das células dos seres vivos, tendo como função basal semi-armazenar as informações necessárias para a construção das proteínas e coordenar o desenvolvimento e funcionamentos dos seres vivos, transmitindo característica hereditárias. Os segmentos do DNA que possuem informações genéticas são denominados genes, descritos por Gregor Mendel (1822-1864), e o restante do DNA é responsável por estruturar e regular essa informação genética. O DNA é constituído por estruturas que se repetem ao longo de seu corpo chamadas de nucleotídeos. Dentro dos seres vivos o DNA não é encontrado como uma cadeia simples e sim como um par de moléculas associadas onde duas cadeias de DNA se unem em uma forma de dupla hélice, na qual os nucleotídeos são unidos por ligações fosfodiéster. A dupla hélice é ligada por pontes de hidrogênio entre as bases, sendo que as quatro bases encontradas no DNA são a Adenina (A), Citosina (C), Guanina (G) e Timina (T). Essas bases são ligadas de maneira que Adenina (A) se liga à Timina (T), e Citosina (C) se liga à Guanina (G) e isso é chamado de par de bases. Com essas ligações em pares, toda informação que uma fita de DNA armazena também é armazenada na outra fita ligada a ela, o que é essencial na duplicação do DNA (WATSON *et al*., 2015). A Figura 1 ilustra a localização e composição do DNA.

Figura 1 - Figura ilustrativa da localização e composição do DNA



Fonte: https://pixabay.com

3.1.2. *Splicing* e Tradução

Durante a síntese de proteínas, o DNA é utilizado para gerar o RNA mensageiro (mRNA), no RNA não há a existência da Timina (T), ela é substituída por outra base, chamada de Uracila (U), assim, após o mRNA ser formado, ele sai do núcleo da célula e é liberado em seu citoplasma até o encontro dos ribossomos, onde ocorre o processo de síntese de proteínas conhecido como tradução (WATSON *et al.*, 2015).

Algumas partes do mRNA não são codantes, conforme descoberto por Phillip Allen Sharp (1944 - atual), isto é, partes do mRNA são descartadas e não participam do processo da tradução. Essas regiões não codantes são denominadas íntrons e os intervalos codantes que irão fazer efetivamente parte do processo de tradução são denominados éxons. Todos os organismos eucariontes possuem íntrons em suas estruturas genéticas e antes do processo de tradução é necessário remover essas partes do mRNA através de um mecanismo molecular chamado spliceossoma no processo conhecido como *splicing* de RNA (CHOW *et al.*, 1977).

Segundo Alberts *et al.* (2017), há padrões nos nucleotídeos quase invariantes no processo de identificação de íntrons, esses padrões são chamados de “sequências consenso”, sendo um destes a

presença das bases GU no início do íntron e AG no seu final. Segue um exemplo ilustrativo da localização dos éxons no RNA em relação ao DNA.

Figura 2 - Figura ilustrativa da localização dos éxons.

Fonte: *National Human Genome Research Institute* (domínio público).

Após o processo de *splicing* ser finalizado no mRNA é iniciado a tradução. O processo de tradução consiste em traduzir uma sequência de nucleotídeos do mRNA em uma sequência de aminoácidos. Esse processo ocorre em trincas, três nucleotídeos específicos para cada aminoácido, chamadas códons (RIDLEY., 2006). A Figura 3 mostra as possíveis combinações de trincas e os correspondentes aminoácidos traduzidos por cada uma. O códon AUG, produtor da Metionina (Met), está destacado pois representa o códon inicial de tradução, e as trincas UAA, UAG e UGA representam os possíveis códons de finalização do processo de tradução.

Figura 3 - Figura da tabela de tradução dos códons.

Second letter

First letter	U	C	A	G	Third letter
U	UUU Phe UUC Phe UUA Leu UUG Leu	UCU Ser UCC Ser UCA Ser UCG Ser	UAU Tyr UAC Tyr **UAA Stop** **UAG Stop**	UGU Cys UGC Cys **UGA Stop** UGG Trp	U C A G
C	CUU Leu CUC Leu CUA Leu CUG Leu	CCU Pro CCC Pro CCA Pro CCG Pro	CAU His CAC His CAA Gln CAG Gln	CGU Arg CGC Arg CGA Arg CGG Arg	U C A G
A	AUU Ile AUC Ile AUA Ile **AUG Met**	ACU Thr ACC Thr ACA Thr ACG Thr	AAU Asn AAC Asn AAA Lys AAG Lys	AGU Ser AGC Ser AGA Arg AGG Arg	U C A G
G	GUU Val GUC Val GUA Val GUG Val	GCU Ala GCC Ala GCA Ala GCG Ala	GAU Asp GAC Asp GAA Glu GAG Glu	GGU Gly GGC Gly GGA Gly GGG Gly	U C A G

Fonte: "O código genético", *OpenStax College*, *Biology* (CC BY 3.0).

3.2. Aprendizado de Máquina

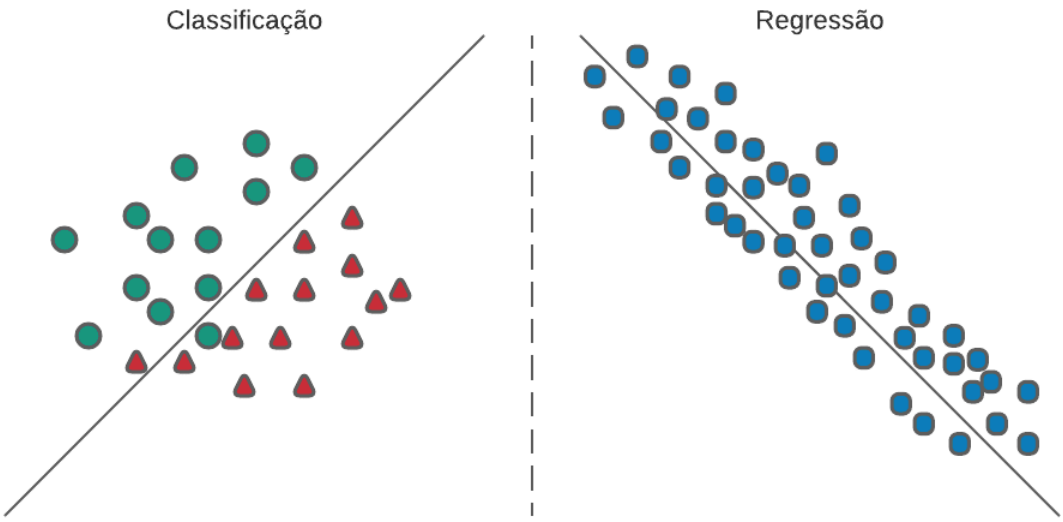
Com o surgimento do computador, a humanidade tem cada vez mais gerado dados e informações digitais, agrupadas e de acesso relativamente fácil. Com o grande avanço tecnológico e a integração de redes de *internet* por todo o mundo, dados sobre qualquer coisa ou situação se tornam cada vez mais comuns e abundantes. O conceito começou a se desenvolver quando entendeu-se que os dados poderiam esconder padrões, e, estes, tinham tamanha proporção que o ser humano já não era mais capaz de analisá-los sem ajuda automatizada. Dessa forma, o Aprendizado de Máquina (AM) ou o ato de ensinar uma máquina ganhou cada vez mais espaço de pesquisa e aplicação. Segundo Arthur Samuel (1959), o AM pode ser definido como: “o campo de estudo que dá aos computadores a habilidade de aprender sem serem explicitamente programados”. O AM pode ser dividido em quatro principais tipos de abordagens: Aprendizado supervisionado, aprendizado não supervisionado, aprendizado por reforço, cada um com suas características e métodos distintos entre si, bem como área de atuação principal e estudos sobre.

O estudo do AM é considerado como uma área multidisciplinar, tendo enfoque não apenas a computação, mas também a matemática, a física e a estatística, sendo que cada uma delas possui importantes contribuições para os modelos atuais de AM. Além disso, o AM pode ser aplicado nas mais diversas áreas de conhecimento, como é o contexto que circunda este trabalho, que aplicou conhecimentos da área da computação, em problemas da biologia e biotecnologia, assim como Rätsch *et al*. (2007), que tentou modelar e prever como o processo de *splicing* acontece.

3.2.1. Aprendizado Supervisionado

O aprendizado supervisionado é uma das formas de aprendizado mais comuns e foi a abordagem utilizada neste trabalho. Ela se baseia na ideia de ensinar à máquina partindo de exemplos já existentes e bem consolidados, trazendo não só a pergunta ou os dados de interesse, mas também o rótulo desses exemplos. Como exemplo deste tipo de aprendizado, podemos citar a tentativa de identificar rostos em fotos tendo à disposição milhares de fotos de diferentes de pessoas, juntamente com a identificação (rótulo) dessas pessoas. Neste tipo de aprendizado, de acordo com Monard e Baranauskas (2003), o modelo é treinado com base em um conjunto de exemplos e seus respectivos rótulos, a fim de possibilitar a construção de um classificador que possa determinar com êxito a classe de exemplos ainda não rotulados. Quando o rótulo esperado dos exemplos é discreto, como o nome da pessoa ou a sua etnia, por exemplo, tem-se um problema de classificação (Figura 4). Quando o rótulo é um valor contínuo, como o valor de venda de uma casa, por exemplo, tem-se um problema de regressão (Figura 4) (CHOWDHURY, 2003). Algumas técnicas famosas de aprendizado supervisionado incluem: regressão linear, regressão logística, redes neurais ou k-vizinhos mais próximos.

Figura 4 - Figura ilustrativa dos problemas de classificação e regressão.



Fonte: Próprios Autores.

3.2.2. Aprendizado Não Supervisionado

O aprendizado não supervisionado é uma técnica de aprendizagem que aprende padrões a partir de dados não rotulados. Diferente do aprendizado supervisionado, onde os dados são classificados previamente por um humano como, por exemplo, uma “maçã” ou “laranja”, o aprendizado não supervisionado utiliza-se de uma auto-organização ou auto-classificação a qual captura padrões presentes nos dados formando agrupamentos (*clusters*) de exemplos semelhantes. Uma forma de medir a distância ou similaridade entre os exemplos faz com que o algoritmo possa encontrar padrões nos dados. Sendo indicado para uso quando deseja-se encontrar valores, classificações ou padrões e não se conhece de maneira clara quais são eles (MCGREGOR *et al*., 2004).

3.2.3. Aprendizado por Reforço

O aprendizado por reforço é uma técnica de aprendizagem que consiste no mapeamento de estados em ações de forma que através do mapeamento dessas ações, busca-se maximizar ou minimizar um determinado valor, conhecido também como recompensa. Não é necessário que o agente (ou modelo a ser treinado) saiba inicialmente das ações a serem tomadas, mas ele deve ao longo do treinamento descobrir quais ações o levam a obter os melhores valores de retorno. Essa técnica também é inspirada em um conceito da psicologia, o qual recompensas e punições são direcionadas a um agente dependendo de suas ações no ambiente. Com a repetição desses passos é esperado que o agente passe a entender ações que evitam punições e maximizam recompensas no ambiente que o cerca (GÉRON, 2019).

3.2.4. Conditional Random Fields

Proposto em 2001 por Lafferty, McCallum e Pereira, o modelo de aprendizado de máquina *Conditional Random Fields* ou CRF, é um modelo de AM supervisionado, que se baseia na probabilidade e na Cadeia Oculta de Markov, para construir um modelo de classificação que utiliza os dados vizinhos à informação de interesse e como eles a afetam para construir uma predição sobre a classificação da mesma. Normalmente aplicado em sequências de texto, o modelo divide e rotula os dados presentes, gerando uma distribuição condicional para os dados da sequência. Atualmente, o CRF tem sido utilizado para a construção de modelos para serem aplicados sobre a figura linguística, buscando padrões de reconhecimento em escrita para diversos fins, como no trabalho de Settles (2004) que propôs um reconhecedor de conhecimento biomédico usando o CRF.

3.2.5. Medidas de desempenho

Após a escolha do modelo a ser utilizado e o treinamento do mesmo é necessário que ele seja avaliado para determinar a sua capacidade de generalização. Normalmente, o modelo é avaliado com base em algumas medidas, como a acurácia e erro, matriz de confusão, precisão, revocação e medida F1.

A medida acurácia é a representação de quantos por cento o modelo acertou dentre todas as suas predições, isso significa que a acurácia é:

$$Acurácia = \frac{QuantidadeDeObservaçõesClassificadasCorretamente}{QuantidadeTotalDeObservações}$$

Já o erro, é definido como:

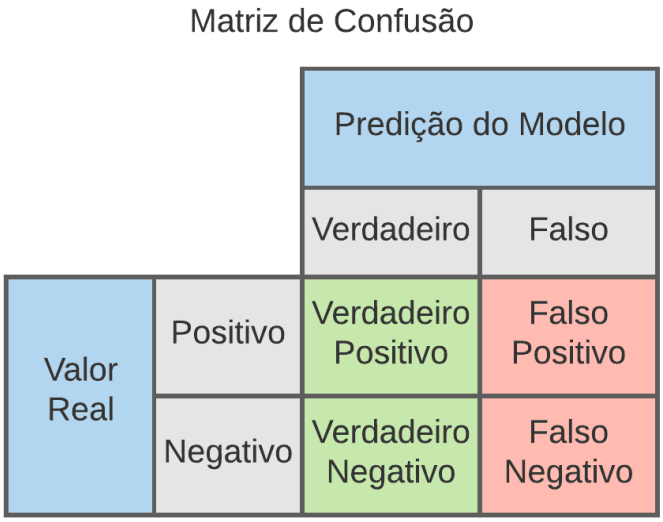
$$Erro = 1 - Acurácia$$

Acurácia e erro são as duas medidas mais básicas de desempenho de um modelo.

A matriz de confusão é uma matriz bidimensional, indexada em uma dimensão pela classe verdadeira de um objeto e na outra pela classe que o classificador atribui (SAMMUT; WEBB, 2017). Dessa forma, a matriz de confusão é formada pelos quatro valores possíveis a seguir, como mostrado na Figura 5:

- Verdadeiro Positivo (ou *True Positive - TP*) - valores classificados como positivos nos dados originais e classificados corretamente como positivos pelo modelo;
- Verdadeiro Negativo (ou *True Negative - TN*) - valores classificados como negativos nos dados originais e classificados corretamente como negativos pelo modelo;
- Falso Positivo (ou *False Positive - FP*) - Valores classificados como negativos nos dados originais e previstos incorretamente como positivos pelo modelo;
- Falso Negativo (ou *False Negative - FN*) - valores classificados como positivos nos dados originais e previstos incorretamente como negativos pelo modelo.

Figura 5 - Figura ilustrativa da Matriz de Confusão.



Fonte: Próprios Autores.

A partir da matriz de confusão, podemos definir algumas medidas de desempenho como:

- Precisão: a precisão é definida contabilizando quantos dos valores classificados pelo modelo como verdadeiro de fato eram verdadeiros. Podemos definir a seguinte fórmula para o cálculo:

$$Precisão = TP/(TP+FP)$$

- Revocação: também chamada de *recall* ou sensibilidade, é calculada considerando quantos dos valores que deveriam ter sido classificados como verdadeiros pelo modelo, realmente foram dados como verdadeiros por ele. Com isso, podemos definir a revocação como:

$$Revocação = TP/(TP+FN)$$

Considerando os valores de precisão e revocação, pode-se calcular a medida F1, que é uma medida harmônica entre as duas, além de ser bastante utilizada para medir o desempenho de um modelo, definida por:

$$F1 = 2*(precisão + revocação)/(precisão + revocação)$$

Neste trabalho, com o auxílio da biblioteca *Scikit Learn*, foram geradas as medidas de precisão, revocação e F1. Porém, para a análise dos resultados apenas a medida F1 foi considerada.

4. Desenvolvimento (Materiais e Métodos)

Inicialmente, este trabalho pretendia usar os resultados gerados por Ferreira (2019), na identificação das regiões codantes, para construir o módulo de tradução dessas regiões para as proteínas correspondentes. Porém, ao estudar os resultados gerados por Ferreira (2019), foram encontrados problemas para determinar as regiões codantes que deveriam ser traduzidas. Isso se deve ao fato de que Ferreira (2019) treinou um modelo para identificar as possíveis regiões que eram íntrons ou não-íntrons, e obteve ótimos valores para a medida F1 do classificador[1], mas não chegou a identificar as regiões de *splicing* na sequência final, após a classificação. Esse processo é necessário, pois, não é possível determinar, em uma sequência desconhecida, as posições de início e fim de íntrons/éxons (regiões de *splicing*). Desta forma, **para modelar o problema de identificação de íntron e éxons como um problema de AM supervisionado, tanto na abordagem proposta por Ferreira (2019) quanto neste trabalho, todas as possibilidades de sequências que começam com GU e terminam com AG formam exemplos de entrada de possíveis íntrons e são submetidas ao classificador.** Durante o treinamento do modelo, as sequências do GenBank já trazem a informação das posições corretas dos íntrons e éxons, mas, quando o modelo treinado está em operação (classificando novos exemplos), após a classificação dessas sequências é necessário remontar as sequências de entrada para determinar as regiões de *splicing* e fazer a tradução das sequências codantes (éxons) para a proteína. Ao utilizar o algoritmo proposto por Ferreira (2019) neste trabalho, não foi possível reconstruir a sequência de entrada e determinar as regiões codantes para a tradução proteica, sendo necessário reformular a abordagem, organizar o código em módulos e reconstruir a base de dados para o treinamento.

A seção 4.1 descreve a construção da base de dados, apresenta a linguagem e as bibliotecas utilizadas na implementação da abordagem. A seção 4.2 descreve os módulos implementados e o funcionamento de cada um deles.

4.1. Construção da base de dados e linguagem utilizada

A base de dados foi construída com base no GenBank, um banco de dados público, amplamente utilizado pelos cientistas das mais diversas áreas que envolve genética, pois possui anotações de sequências de nucleotídeos dos mais diversos seres vivos. Mantido e gerenciado pelo *National Center for Biotechnology Information*, o banco conta com milhares de sequências de diversos fungos e animais. Para este trabalho, foram selecionados dois conjuntos de sequências: um contendo sequências de DNA do fungo *Colletotrichum* e outro contendo sequências de DNA do fungo *Diaporthe*. Para a escolha deste conjunto, foram considerados os estudos feitos por Ferreira (2019) e a aplicação dos resultados sobre eles. A base de dados de Ferreira (2019) contava com um total de 5209 amostras (sequências) do fungo

[1] Medida F1 = 0.96 para o modelo treinado com o fungo Colletotrichum e 0.84 para o modelo treinado com o fungo Diaporthe.

Colletotrichum, armazenadas em um arquivo de 12,3 MB e 1821 amostras (sequências) do fungo *Diaporthe*, armazenadas em um arquivo de 5,4 MB, como mostra a Tabela 1.

Tabela 1. Descrição das bases de dados para os fungos *Colletotrichum* e *Diaporthe* construídas por Ferreira (2019).

Fungo	**Proteína**	**Quantidade de amostras**	**Tamanho em MB do arquivo**
Colletotrichum	Actina	5.209	12,3
Diaporthe	Beta Tubulina	1.821	5,4
Total		7.030	17,7

Fonte: Próprios Autores.

Com as atualizações feitas no GenBank com o passar do tempo, foi possível aumentar o tamanho da base de dados para este trabalho. Foram recuperadas 7005 amostras (sequências) do *Colletotrichum*, em um arquivo de 16,2 MB e 2109 e 155 amostras (sequências) do *Diaporthe,* para as proteínas beta tubulina e actina, em um arquivo de 6,2 MB e 400 KB respectivamente, como mostra a Tabela 2.

Tabela 2. Descrição das bases de dados para os fungos *Colletotrichum* e *Diaporthe* construídas neste trabalho.

Fungo	**Proteína**	**Quantidade de amostras**	**Tamanho em MB do arquivo**
Colletotrichum	Actina	7.005	16,2
Diaporthe	Beta Tubulina	2.109	6,2
Diaporthe	Actina	155	0,4
Total		9.114	22,8

Fonte: Próprios Autores.

Para recuperar as sequências do *GenBank*, foram usadas as seguintes *strings* (sequência de texto) de busca:

- Colletotrichum (actina): *colletotrichum actin partial CDS;*

- Diaporthe (beta tubulina): *diaporthe beta-tubulin partial CDS;*
- Diaporthe (actina): *diaporthe actin partial CDS;*

Foram selecionadas apenas as sequências do fungo *Colletotrichum* que correspondem à proteína actina e o para o fungo *Diaporthe* foram selecionadas as sequências que correspondem à proteína beta tubulina e actina[2]. Além disso, foi adicionada a restrição de "*partial CDS"*, que filtra as sequências com bases degeneradas, isto é, contém uma possível ambiguidade em uma ou mais posições da sequência que o pesquisador responsável pelo depósito da sequência no GenBank não foi capaz de decifrar.

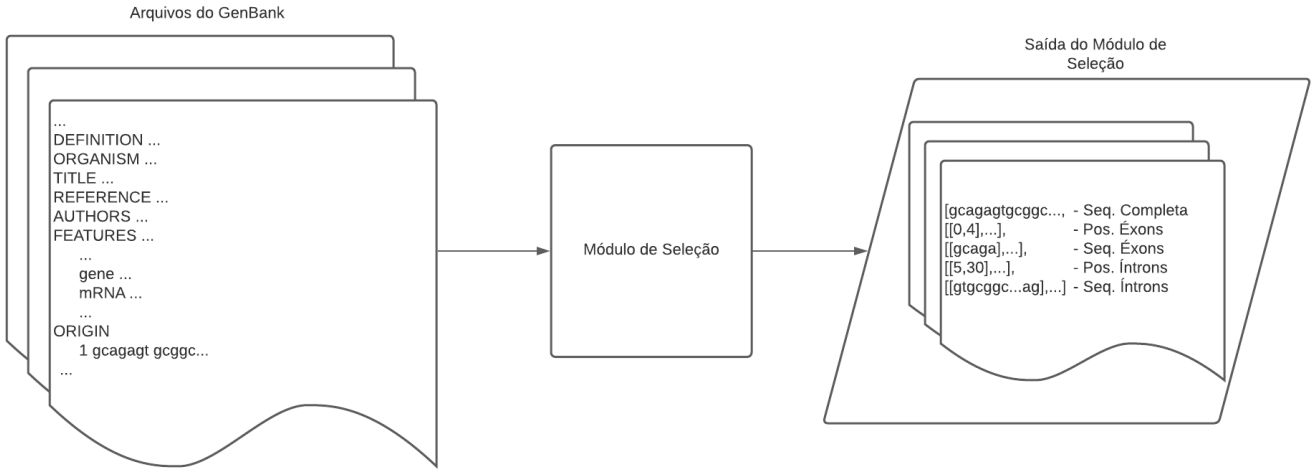
Para o desenvolvimento dos algoritmos, módulos e modelos, a linguagem escolhida foi *Python*, fazendo uso das bibliotecas *sklearn*, *crfsuite*, *pickle* e *random*. As bibliotecas *sklearn* e *crfsuite* contam com funções, modelos e técnicas de treinamento já implementados, bem como funções de métricas. Foi feito uso de ambas para o treinamento e exposição dos resultados baseado no modelo de AM do *Conditional Random Fields*. A biblioteca *pickle* foi utilizada para o gerenciamento dos arquivos gerados pelos módulos do algoritmo, isto é, leitura e gravação, e a biblioteca *random* foi usada para auxiliar em atividades que necessitassem de escolhas aleatórias.

4.2. Estruturação do Código em Módulos

Foram desenvolvidos quatro módulos de implementação com intuito de tornar o código mais organizado e manutenível. Assim como no trabalho de Ferreira (2019), a abordagem empregada neste projeto foi usar o modelo de AM baseado no CRF. Os três módulos principais resultam no modelo completamente treinado e pronto para classificar novas sequências ainda não apresentadas ao modelo. O último módulo é o módulo de aplicação, onde o modelo já treinado recebe uma entrada de dados e devolve como resposta a posição dos íntrons, dos éxons, assim como a sequência final com o processo de *splicing* realizado e as possíveis traduções da sequência codante para proteína.

[2] As sequências do fungo *Diaporthe* para a proteína actina foram selecionadas após a verificação de que as medidas de desempenho dos algoritmos treinados anteriormente (fungo *Colletotrichum* proteína actina e fungo *Diaporthe* proteína beta tubulina) eram muito diferentes para as duas proteínas. Após a consulta com um especialista da área de biotecnologia, o mesmo detectou que essas diferenças poderiam ter sido geradas porque as regiões do DNA selecionadas para cada proteína eram diferentes, gerando inclusive, sequências de tamanhos mais curtos ou mais longos dependendo desse fator.

Figura 6 - Funcionamento detalhado do módulo de seleção.

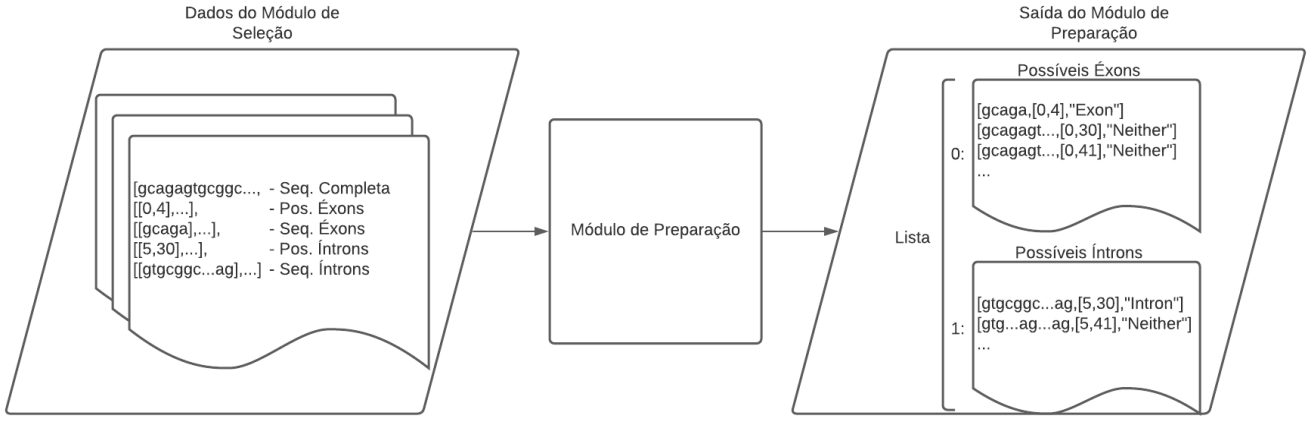


Fonte: Próprios Autores.

A estrutura lógica do algoritmo completo e suas fases até o treinamento é estruturada conforme mostra a Figura 9. Inicialmente, os dados recuperados/escolhidos do GenBank são guardados em arquivos à parte e são usados para o primeiro módulo do algoritmo, o módulo de seleção. Neste primeiro módulo, os dados do GenBank são selecionados e, para cada uma das sequências, é montada uma lista contendo a sequência completa, uma lista das posições de cada éxon da sequência, uma lista contendo cada uma das sequências de éxons da sequência, uma lista das posições de cada íntron e, por último, uma lista contendo cada uma das sequências de íntrons. O resultado parcial é salvo em um novo arquivo, que, por sua vez, é a entrada do segundo módulo, como é mostrado na Figura 6.

O segundo módulo, é o módulo de preparação da base. Os dados de interesse das sequências recuperadas já terão sido selecionadas no módulo anterior e agora serão preparados para comporem a base de dados para o treinamento do CRF. Neste módulo está o principal diferencial deste trabalho em relação ao trabalho implementado por Ferreira (2019). Ao analisar a base de dados de entrada utilizada por Ferreira (2019), notou-se um desbalanceamento significativo entre a quantidade de sequências nas classes-alvo do classificador. Naquele trabalho, havia apenas duas classes para as quais o classificador poderia classificar uma sequência: a classe “íntron” e a classe “not-íntron”. Como os exemplos criados naquele trabalho para compor a base de dados foram derivados dos padrões GU no início do íntron e AG no seu final, todas as possibilidades de combinação entre GU e AG geradas foram colocadas na base, mas somente os íntrons verdadeiros faziam parte da classe “intron”, enquanto todas as outras, faziam parte da classe “not-íntron”. Isso fez com que a classe “not-íntron” tivesse um número de exemplos muito maior do que a classe “íntron”, desvirtuando assim, o resultado do modelo (todas as sequências novas apresentadas ao classificador, eram classificadas como “not-íntron”).

Figura 7 - Funcionamento detalhado do módulo de preparação.



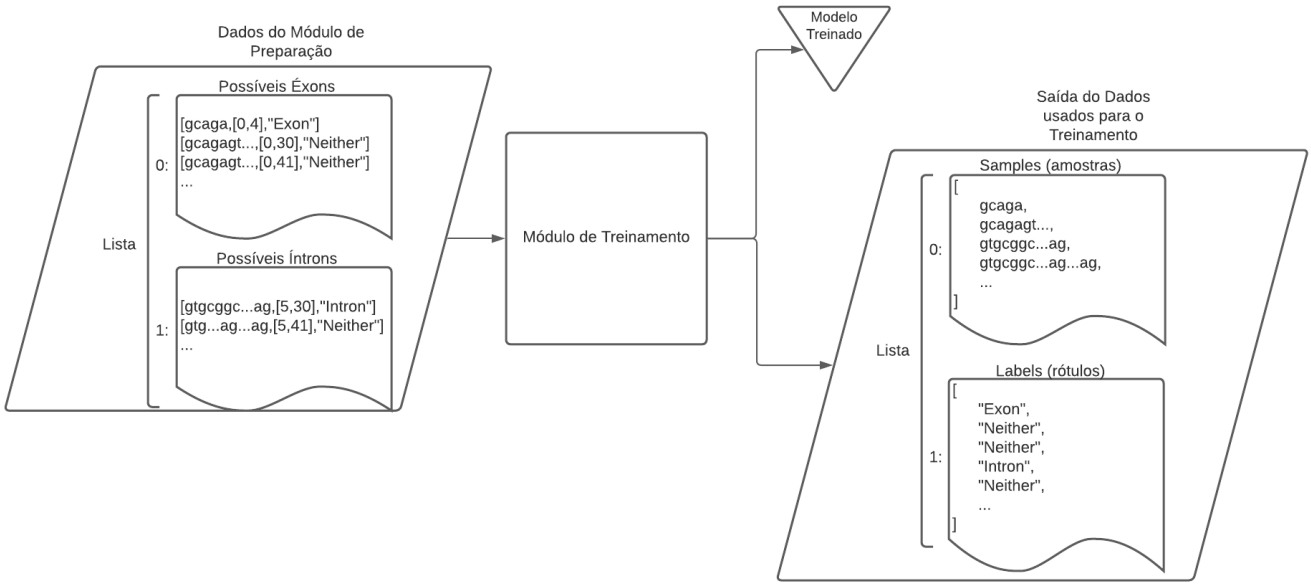
Fonte: Próprios Autores.

Por isso, neste trabalho a base de dados foi estruturada de forma que os exemplos são classificados em três possíveis classes-alvo: íntron, éxon ou *neither* (nenhuma das opções, traduzido do inglês), na tentativa de encontrar possíveis padrões também nos éxons (regiões codantes). As sequências que são classificadas como íntron ou éxon, já foram obtidas no módulo anterior, as sequências que irão compor a base como *neither* (uma mistura de íntron e éxon) são criadas neste módulo da seguinte forma: são geradas todas as combinações de possíveis íntrons e possíveis éxons de cada uma das sequências, sendo que para isso leva-se em conta os padrões de GU no início do íntron e AG no final. As sequências que correspondem a íntrons e éxons verdadeiramente são rotuladas como tal, como é exemplificado na Figura 7, as restantes são rotuladas como *neither*.

Ao longo do projeto, constatou-se que a base de treinamento rotulada alcançava pelo menos cinco vezes o tamanho inicial da base de dados dos fungos de entrada, recuperados no GenBank.

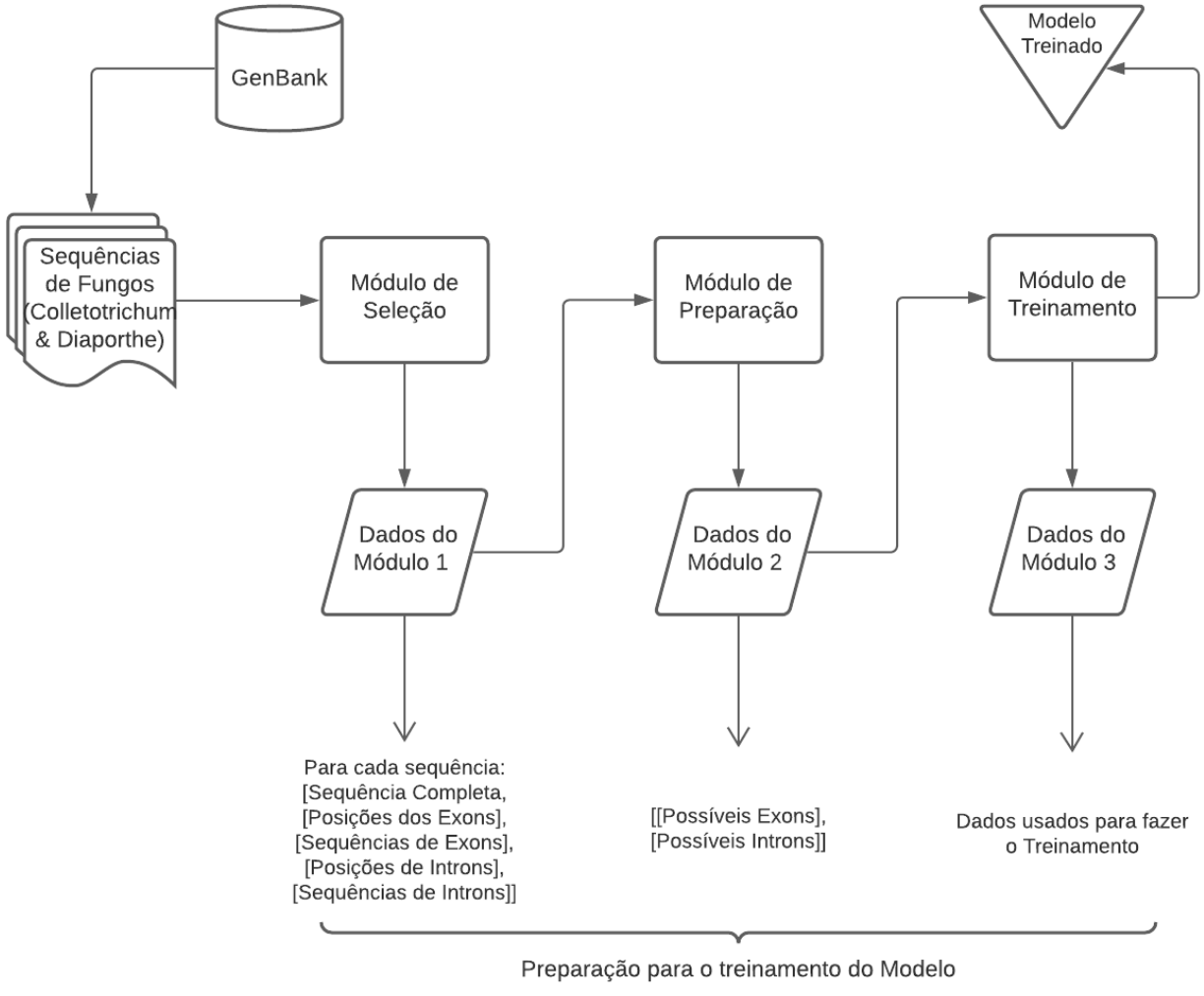
Por fim, o terceiro módulo, chamado de módulo de treinamento, é alimentado com os dados gerados anteriormente. As sequências que correspondem a íntrons e éxons verdadeiramente são retiradas, e entre as sequências restantes, são escolhidas aleatoriamente uma quantidade de exemplos igual a média simples entre a quantidade de íntrons e éxons para compor a classe *neither*, mantendo o balanceamento no número de exemplos classificados em cada classe. Foram consideradas como *features* para a entrada do treinamento do modelo apenas a sequência inteira do éxon ou íntron. O modelo então é treinado e salvo em um último arquivo intermediário que pode ser lido e usado para predições, como mostra a Figura 8.

Figura 8 - Funcionamento detalhado do módulo de Treinamento.



Fonte: Próprios Autores.

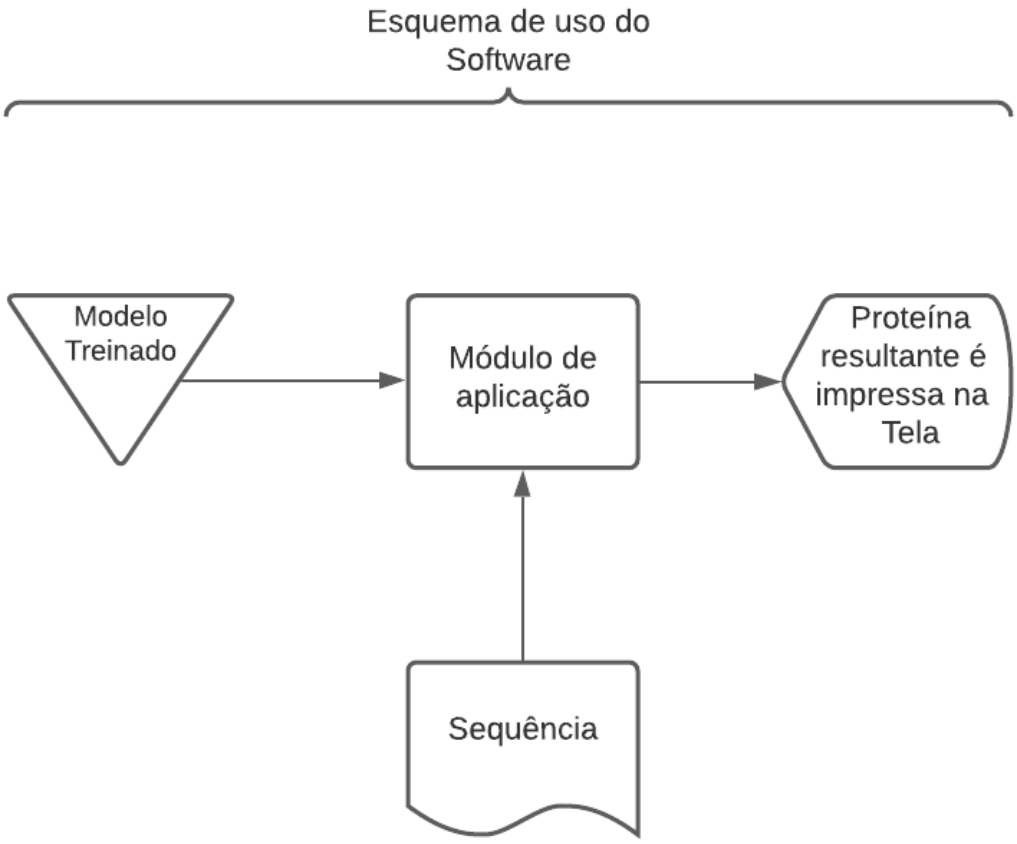
Figura 9 - Modelo de funcionamento para o treinamento do modelo.



Fonte: Próprios Autores.

Dessa forma, o modelo treinado é usado no módulo de aplicação, onde novos exemplos de dados são fornecidos e o modelo é usado para predizer as posições de *splicing* da sequência fornecida, resultando em uma sequência final que passa por um processo de tradução para proteína e os resultados obtidos são apresentados (Figura 10). Para o processo de tradução, foi usado um dicionário de trincas para proteína dentro de uma iteração nas trincas da sequência para gerar a proteína resultante. Além disso, foram consideradas as possibilidades da sequência resultante ter a decodificação em proteína iniciando na posição original da sequência, na posição +1 e na posição +2, produzindo assim todas as possíveis traduções da sequência.

Figura 10 - Esquema de uso do modelo treinado.



Fonte: Próprios Autores.

Por fim, foi criado um algoritmo para testar a precisão real da abordagem, tentando, a partir das classificações individuais de cada exemplo, identificar as regiões de *splicing* nas sequências inteiras obtidas do GenBank. Chamamos esse processo de **avaliação real da classificação resultante do CRF** na obtenção das regiões de *splicing*. O algoritmo recupera todas as sequências do GenBank, monta a sequência pós *splicing* e salva como a resposta correta, e, então, usa o modelo para predizer as regiões dos

íntrons - visto que esta classe obteve o maior valor para a medida F1 nos modelos treinados. Depois, o algoritmo monta a sequência resultante, removendo da sequência do GenBank os íntrons classificados como tal pelo modelo treinado e compara a sequência resultante, caracter por caracter, com a sequência correta final, vinda do GenBank. Se a comparação de todos os caracteres for idêntica, o resultado é dado como correto, caso contrário, é dado como incorreto. No final, é tomada a quantidade de acertos divididos pela quantidade de exemplos, ou seja, calcula-se a acurácia, e esta é retornada como resposta do algoritmo.

Todo o código desenvolvido, seus módulos, arquivos do GenBank utilizados e de configurações estão disponíveis em: https://github.com/GustavoHCruz/RNACodingRegions.

5. Resultados e Discussões

Primeiramente foi utilizada a base de dados do GenBank obtida por Ferreira (2019), que contava com 5209 amostras para o fungo do *Colletotrichum*, mas, com a base de dados formada pelas três classes: íntrons, éxons e *neither.* A Tabela 3 mostra os resultados obtidos da medida F1 para cada classe nesse caso:

Tabela 3 - Medidas obtidas pelo modelo treinado para o fungo *Colletotrichum* e proteína actina.

Rótulo	**Quantidade**	**% de exemplos na base**	**Medida F1**
Intron	9.649	26,59%	0,94
Exon	14.541	40,07%	0,79
Neither	12.095	33,33%	0,78
Total	36.285	100%	

Fonte: Próprios Autores.

Para o fungo *Diaporthe*, que contava com 1821 amostras, os resultados obtidos para a medida F1 nas três classe são mostrados na Tabela 4:

Tabela 4 - Medidas obtidas pelo modelo treinado para o fungo *Diaporthe* e proteína beta tubulina.

Rótulo	**Quantidade**	**% de exemplos na base**	**Medida F1**
Intron	5.990	28,46%	0,84
Exon	8.040	38,20%	0,67
Neither	7.015	33,33%	0,21
Total	21.045	100%	

Fonte: Próprios Autores.

Tomando como referência o desempenho superior do modelo na proteína actina, e com instruções de profissionais da área da biotecnologia, optou-se por testar ambos os fungos com ela, usando a base de dados atualizada, isto é, com a consulta que retornou 7005 sequências do *Colletotrichum* e 155 sequências do *Diaporthe*, ambos para a proteína actina. Foram obtidos os seguintes resultados:

Tabela 5 - Medidas obtidas pelo modelo treinado para o fungo *Colletotrichum* usando a base de dados atualizada para a proteína actina.

Rótulo	Quantidade	% de exemplos na base	Medida F1
Intron	13.017	26,55%	0,96
Exon	19.663	40,11%	0,84
Neither	16.340	33,33%	0,75
Total	49.020	100%	

Fonte: Próprios Autores.

Tabela 6 - Medidas obtidas pelo modelo treinando para o fungo *Diaporthe* usando a base de dados atualizada para a proteína actina.

Rótulo	Quantidade	% de exemplos na base	Medida F1
Intron	308	26,64%	0,79
Exon	463	40,05%	0,68
Neither	385	33,33%	0,27
Total	1.156	100%	

Fonte: Próprios Autores.

Os resultados obtidos para o *Colletotrichum* apresentaram uma pequena melhoria geral em todas as precisões, devido ao aumento da base de dados usada para o treinamento. Para o fungo *Diaporthe*, os resultados foram pouco descritivos já que a base de dados contava com uma quantidade muito pequena de dados, porém, se manteve com uma medida F1 igual ao modelo anterior. Apesar da mudança na proteína utilizada para obter as sequências, a medida F1 continua sendo inferior no *Diaporthe* em relação ao *Colletotrichum*.

Tabela 7 - Resultados da avaliação real da classificação resultante do CRF na base de dados de Ferreira (2019).

Fungo	Proteína	Acurácia
Colletotrichum	Actina	83,30%
Diaporthe	Beta Tubulina	27,30%

Fonte: Próprios Autores.

Após a obtenção das medidas descritas, obtidas diretamente durante o treinamento do modelo, foi realizada a **avaliação real da classificação resultante do CRF** na obtenção das regiões de *splicing*. Utilizando a base de dados de Ferreira (2019), o modelo obteve uma precisão de 83,3% em sequências do *Colletotrichum* (para actina) e de 27,3% para sequências do *Diaporthe* (para beta tubulina), resultados descritos na Tabela 7.

Com a base atualizada, selecionando dessa vez as sequências da proteína actina para os dois fungos, obteve-se a precisão de 82,5% para o *Diaporthe* e 100% para o *Colletotrichum*, como mostram os resultados descritos na Tabela 8.

Tabela 8 - Resultados da avaliação real da classificação resultante do CRF na base de dados atualizada

Fungo	**Proteína**	**Acurácia**
Colletotrichum	Actina	82,50%
Diaporthe	Actina	100%

Fonte: Próprios Autores.

Além desta comparação, que foi capaz de validar o modelo na prática, foi feita uma segunda análise, usando a ferramenta BLAST[3] do GenBank, que foi capaz de identificar com uma compatibilidade de 95% nas sequências resultantes do modelo do *Colletotrichum* para a proteína actina. Considerando a baixa quantidade de exemplos que foram usados para o treinamento do modelo do *Diaporthe* e a resposta inconclusiva que foi obtida, não foram feitas comparação com ele.

[3]O BLAST é um algoritmo usado na comparação de informações biológicas, como sequências de aminoácidos ou nucleotídeos de sequências de DNA.

6. Conclusões

Este trabalho reestruturou a abordagem proposta por Ferreira (2019) para a identificação de regiões codantes de fungos filamentosos. Primeiramente, o algoritmo foi reorganizado em módulos, para facilitar a manutenção futura do mesmo e permitir que as sequências pudessem ser remontadas após o resultado da classificação do CRF. Depois disso, a base de dados obtida do GenBank foi reconstruída e atualizada, incorporando novas sequências e três classes possíveis de resultado de classificação: intron, exon e neither. Além disso, um módulo de avaliação real da classificação resultante do CRF foi implementado.

Com base nos resultados obtidos podemos concluir que a abordagem se mostrou efetiva para o fungo *Colletotrichum* com a proteína actina (medida F1 acima de 0,94 e precisão real acima de 82%). Já para o fungo *Diaporthe* as medidas não foram tão boas quando testadas as sequências de beta tubulina (medida F1 de 0,84 e precisão real de 27,3%). Para o *Diaporthe*, as sequências de beta tubulina deram resultados inferiores aos esperados, apesar dos resultados mostrarem um alto valor para a medida F1, enquanto as sequências de actina foram escassas (apenas 155) e o modelo possivelmente não foi capaz de generalizar um padrão, deixando os resultados para essa proteína neste fungo inconclusivos o que pode explicar a taxa de acerto de 100%. Para o *Colletotrichum*, os resultados foram satisfatórios para sequências de actina, o que garantiu uma boa precisão ao identificar as regiões de *splicing* e remontar corretamente uma sequência para obter as regiões de *splicing*.

Podemos concluir que o *Colletotrichum* mantêm uma boa taxa de acertos para as sequências completas, enquanto que o *Diaporthe* tem uma baixa precisão para beta tubulina e um resultado inconclusivo para a actina. A baixa precisão na beta tubulina provavelmente se deve às cadeias de gene maiores, com cerca de duas a cinco vezes maiores que as sequências gênicas da actina no *Colletotrichum*, o que diminuiu o valor de F1. Além disso, essas sequências maiores possuem mais íntrons e éxons do que a actina, o que resultava em uma maior dificuldade de acerto da cadeia resultante correta, pois a métrica utilizada para validação do resultado não permitia que houvesse nenhuma variação de caractere entre as sequências comparadas para que fosse considerada correta.

Mesmo com a detecção dos problemas na abordagem adotada por Ferreira e a reestruturação do algoritmo e da base de dados feita neste trabalho, todas as etapas previstas no projeto original forma cumpridas, com exceção da etapa A6 - Propor uma forma de estruturar a base de dados com base em cada nucleotídeo, que foi considerada inadequada em termos de quantidade de processamento, após os testes realizados neste trabalho.

Uma limitação deste trabalho é que o algoritmo não lida bem com sequências degeneradas, ou seja, quando uma sequência têm um ou mais nucleotídeos que não foram identificados (condição esta que é filtrada na busca do banco de dados). Caso o usuário do sistema possua uma sequência desta forma, não será possível realizar a tradução, porém, a análise pelo modelo ainda será feita. Uma abordagem para

lidar com as sequências que possuem bases degeneradas, o que segundo o especialista na área, é bastante comum de ocorrer na prática, é uma sugestão para trabalhos futuros.

Além disso, sugere-se como trabalhos futuros uma abordagem utilizando um algoritmo de AM diferente do CRF, como por exemplo, uma rede neural, estrutura que tem demonstrado ter bons resultados em análises de sequências de texto. Além disso, procurar por sequências de actina de outros fungos e testar um possível padrão genérico baseado nesta proteína (a principal sugestão é a actina) para a identificação das regiões de *splicing* também pode ajudar a entender melhor os padrões com base nas proteínas. Também pode ser interessante testar outras modelagens para o problema e outras formas de validar os resultados, como por exemplo, o próprio algoritmo BLAST.

Sugere-se também a criação de um serviço WEB com interface e atrelado a um servidor, que possa ser capaz de proporcionar o uso do software de identificação de regiões codantes em fungos filamentosos por qualquer pessoa interessada em usufruir da ferramenta.

Com este trabalho foi possível desenvolver a maturidade para o entendimento da área da AM aplicada na identificação de íntrons e éxons, além de aprimorar as habilidades de implementação de técnicas de AM, raciocínio lógico, aplicação de modelos e escrita científica.

7. Bibliografia

ALBERTS, B. *et al.* Biologia molecular da célula. 6. ed. [S.1]: Artmed Editora, 2017. ISBN 9788582714225.

CHOW, L.T., ROBERTS, J.M., LEWIS, J.B., BROKER, T.R. (1977) A map of cytoplasmic RNA transcripts from lytic adenovirus type 2, determined by electron microscopy of RNA:DNA hybrids.

CHOWDHURY, G. G. Natural language processing. Annual review of information science and technology, Wiley Online Library, v. 37, n. 1, p. 51–89, 2003.

FERREIRA, GUSTAVO L. F. Identificando regiões não codantes em sequências de DNA utilizando técnicas de aprendizado de máquina. 54p. TCC (Bacharelado). Universidade Estadual de Maringá. Maringá, 2019.

GÉRON, A. Mãos à obra: Aprendizado de máquina com scikit-learn & tensorflow. Alta Books, 2019. Disponível em: https://books.google.com.br/books?id=Z0mvDwAAQBAJ

LAFFERTY, J.; MCCALLUM, A.; PEREIRA, F. C. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. 2001.

L. S. ARTHUR. Some Studies in Machine Learning Using the Game of Checkers. IBM Journal of Research and Development. 1959. doi:10.1147/rd.441.0206.

MAKAL, S. & OZYILMAZ, L. & PALAVAROGLU, S.. (2008). Neural Network Based Determination of Splice Junctions by ROC Analysis. Proc. World Acad. Sci. Eng. Technol.. 43.

MCGREGOR, A. *et al*. Flow clustering using machine learning techniques. In: BARAKAT, C.; PRATT, I. (Ed.). Passive and Active Network Measurement. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004. p. 205–214. ISBN 978-3-540-24668-8.

MONARD, M. C.; BARANAUSKAS, J. A. Conceitos sobre aprendizado de máquina. Sistemas inteligentes-Fundamentos e aplicações, v. 1, n. 1, p. 32, 2003.

RÄTSCH, G. et al. Improving the caenorhabditis elegans genome annotation using machine learning. PLoS Computational Biology, Public Library of Science, v. 3, n. 2, p. e20, 2007.

RIDLEY, MARK. Evolução. 3. ed. Artmed, 2006.

SETTLES, B. (2004). "Biomedical named entity recognition using conditional random fields and rich feature sets". Proceedings of the International Joint Workshop on Natural Language Processing in Biomedicine and its Applications. pp. 104–107.

WATSON, JAMES D. *et al*. Biologia Molecular do Gene. 7. ed. Artmed, 2015.