UNIVERSIDADE FEDERAL DE SÃO CARLOS

CENTRO DE CIÊNCIAS EXATAS E DE TECNOLOGIA

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

APLICANDO PRINCÍPIOS DE APRENDIZADO DE MÁQUINA NA CONSTRUÇÃO DE UM BIOCURADOR AUTOMÁTICO PARA O GENE ONTOLOGY (GO)

LAURENCE RODRIGUES DO AMARAL

ORIENTADOR: PROF. DR. ESTEVAM RAFAEL HRUSCHKA JÚNIOR

São Carlos – SP

Outubro/2013

UNIVERSIDADE FEDERAL DE SÃO CARLOS

CENTRO DE CIÊNCIAS EXATAS E DE TECNOLOGIA

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

APLICANDO PRINCÍPIOS DE APRENDIZADO DE MÁQUINA NA CONSTRUÇÃO DE UM BIOCURADOR AUTOMÁTICO PARA O GENE ONTOLOGY (GO)

LAURENCE RODRIGUES DO AMARAL

Tese apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de São Carlos, como parte dos requisitos para a obtenção do título de Doutor em Ciência da Computação, área de concentração: Inteligência Artificial

Orientador: Prof. Dr. Estevam Rafael Hruschka Júnior

São Carlos – SP

Outubro/2013

Ficha catalográfica elaborada pelo DePT da Biblioteca Comunitária/UFSCar

A485ap

Amaral, Laurence Rodrigues do.
Aplicando princípios de aprendizado de máquina na construção de um biocurador automático para o Gene Ontology (GO) / Laurence Rodrigues do Amaral. -- São Carlos : UFSCar, 2014.
109 f.

Tese (Doutorado) -- Universidade Federal de São Carlos, 2013.

1. Ciência da computação. 2. Aprendizado de máquina sem-fim. 3. Ontologia genética. 4. Composição de classificadores. 5. Biocuragem. I. Título.

CDD: 004 (20ª)

Universidade Federal de São Carlos

Centro de Ciências Exatas e de Tecnologia

Programa de Pós-Graduação em Ciência da Computação

"Aplicando Princípios de Aprendizado de Máquina na Construção de um Biocurador Automático para o Gene Ontology (GO)"

Laurence Rodrigues do Amaral

Tese de Doutorado apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de São Carlos, como parte dos requisitos para a obtenção do título de Doutor em Ciência da Computação.

Membros da Banca:

Prof. Dr. Estevam Rafael Hruschka Junior
(Orientador - DC/UFSCar)

Profa. Dra. Heloisa de Arruda Camargo
(DC/UFSCar)

Profa. Dra. Maria do Carmo Nicoletti
(PPG-CC/ UFSCar)

Prof. Dr. André Carlos Ponce de Leon Ferreira de Carvalho
(ICMC/USP)

Profa. Dra. Gina Maira Barbosa de Oliveira
(FACOM/UFU)

São Carlos
Outubro/2013

Dedico esse trabalho à minha esposa Kyara e à minha filha Manuela

AGRADECIMENTOS

Como costumo dizer em seções de agradecimentos, mais uma etapa foi atingida nessa caminhada que é a vida. Etapa complexa, cheia de desafios, alegrias, tristezas, conquistas, perdas e cheia de aprendizado. Desta forma, gostaria de agradecer primeiramente a Deus por não me deixar faltar saúde, responsabilidade, garra e acima de tudo esperança na busca pelo sonho chamado doutorado. Também gostaria de agradecer à minha esposa Kyara, companheira de todas as horas, boas ou ruins, que comprou comigo este sonho e também lutou por ele até o fim, sem uma única vez, duvidar se havíamos tomado a decisão correta. A cada dia que passa, confirmo o que eu já sabia, você é uma mulher forte e guerreira que não desiste dos nossos sonhos. À nossa filha Manuela que soube entender que o Papai precisava trabalhar até mais tarde ou nos finais de semana e que não poderia brincar naquele momento. Kyara e Manuela, prometo recompensar todo o tempo abdicado a vocês daqui para frente. Kyara e Manuela muito obrigado. Ao Prof. Estevam Hruschka, professor exemplar e pesquisador brilhante, por caminhar ao meu lado durante estes quatro anos, me ensinando, com grande competência, como um professor/pesquisador deve agir, sempre com ética, responsabilidade e acima de tudo, primando pela máxima qualidade em suas pesquisas e atenção aos orientados. Não posso esquecer também dos meus pais, Ademir e Laurita, meus sogros João e Líbia pelo apoio. A todos os amigos da Universidade Federal de Goiás (UFG - Jataí) pelo apoio nesta caminhada, principalmente no seu início e aos amigos da Universidade Federal de Uberlândia (UFU - Patos de Minas) pelo apoio neste final. Ao Prof. Foued Espindola (INGEB/UFU) por ceder o parque tecnológico do Laboratório de Bioinformática do LaBiBi/UFU (Laboratório de Bioinformática) utilizados na execução dos experimentos e pelo apoio dado. Muito obrigado a todos vocês!

Ostras felizes não fazem pérolas. Pessoas felizes não sentem a necessidade de criar. O ato criador, seja na ciência ou na arte, surge sempre de uma dor. Não é preciso que seja uma dor doída. Por vezes a dor aparece como aquela coceira que tem o nome de curiosidade.

Rubem Alves

RESUMO

Nos dias atuais, a quantidade de dados biológicos disponibilizados por universidades, hospitais e centros de pesquisa tem aumentado de forma exponencial, devido ao emprego da bioinformática, através do desenvolvimento de métodos e técnicas computacionais avançados, e de técnicas de *high-throughput*. Devido a esse significativo aumento na quantidade de dados disponibilizados, gerou-se a necessidade da criação de novas estratégias para captura, armazenamento e principalmente análise desses dados. Devido a esse cenário, um novo campo de trabalho e pesquisa vem surgindo, chamado *biocuragem*. A biocuragem está se tornando parte fundamental na pesquisa biomédica e biológica, e tem por principal função estruturar e organizar a informação biológica, tornando-a legível e acessível a homens e computadores. Buscando prover um rápido e confiável entendimento de novos domínios, diferentes iniciativas estão sendo propostas, tendo no *Gene Ontology* (GO) um dos seus principais exemplos. O GO se destaca mundialmente sendo uma das principais iniciativas em bioinformática, cuja principal meta é padronizar a representação dos genes e seus produtos, provendo interconexões entre espécies e bancos de dados. Dessa forma, objetiva-se com essa pesquisa propor uma arquitetura computacional que utiliza princípios de aprendizado de máquina sem-fim para auxiliar biocuradores do GO na tarefa de classificação de novos termos, tarefa essa, totalmente manual. A arquitetura proposta utiliza aprendizado semi-supervisionado combinando diferentes classificadores na rotulação de novas instâncias do GO. Além disso, essa pesquisa também tem por objetivo a construção de conhecimento de alto-nível na forma de simples regras SE-ENTÃO e árvores de decisão. Esse conhecimento gerado pode ser utilizado pelos biocuradores do GO na busca por padrões importantes presentes nos dados biológicos, revelando informações concisas e relevantes sobre o domínio da aplicação.

Palavras-chave: Aprendizado de Máquina Sem-Fim, *Gene Ontology*, Composição de Classificadores, Biocuragem

ABSTRACT

Nowadays, the amount of biological data available by universities, hospitals and research centers has increased exponentially due the use of bioinformatics, with the development of methods and advanced computational tools, and high-throughput techniques. Due to this significant increase in the amount of available data, new strategies for capture, storage and analysis of data are necessary. In this scenario, a new research area is developing, called biocuration. The biocuration is becoming a fundamental part in the biological and biomedical research, and the main function is related with the structuration and organization of the biological information, making it readable and accessible to mens and computers. Seeking to support a fast and reliable understanding of new domains, different initiatives are being proposed, and the Gene Ontology (GO) is one of the main examples. The GO is one the main initiatives in bioinformatics, whose main goal is to standardize the representation of genes and their products, providing interconnections between species and databases. Thus, the main objective of this research is to propose a computational architecture that uses principles of never-ending learning to help biocurators in new GO classifications. Nowadays, this classification task is totally manual. The proposed architecture uses semi-supervised learning combining different classifiers used in the classification of new GO samples. In addition, this research also aims to build high-level knowledge in the form of simple IF-THEN rules and decision trees. The generated knowledge can be used by the GO biocurators in the search for important patterns present in the biological data, revealing concise and relevant information about the application domain.

Keywords: Never-Ending Language Learner, Gene Ontology, Combining classifiers, Biocuration

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

Capítulo 1

INTRODUÇÃO

Atualmente a quantidade de dados biológicos disponibilizados pelas universidades, hospitais e laboratórios de pesquisa tem crescido de forma exponencial, amparado pelo crescimento da bioinformática e pelo desenvolvimento de métodos e técnicas computacionais avançados. Esse significativo aumento na quantidade de dados biológicos disponibilizados, gerou a necessidade de novas estratégias de captura, armazenamento e análise. Além disso, o número e o alcance das bases de dados científicas tem crescido nesses últimos anos, criando um novo campo de trabalho e pesquisa chamado *biocuragem* (HOWE et al., 2008; SALIMI; VITA, 2006).

A biocuragem está se tornando parte essencial da pesquisa biológica e biomédica. Ela pode ser definida como a atividade de estruturar e organizar a informação biológica, tornando-a acessível a humanos e computadores (HOWE et al., 2008; BATEMAN, 2010). Mais precisamente, a biocuragem envolve a análise, interpretação e a integração das informações biológicas, interconectando dados de pesquisas para a criação de um ambiente biológico comum. Essa integração traz duas importantes contribuições: 1) A expansão do acesso a esses dados dentro da comunidade científica e 2) Crescimento no número de descobertas científicas devido ao aumento no número de análises computacionais (BURGE et al., 2012).

A biocuragem requer habilidades e conhecimentos avançados em pesquisa científica e também em Computação, pois os biocuradores trabalham corriqueiramente com sistemas gerenciadores de bancos de dados, sistemas operacionais e linguagens de programação. Esses biocuradores são profissionais responsáveis pela aplicação de métodos e ferramentas na manipulação desses dados. Além disso, eles também são responsáveis pelo gerenciamento dessas ferramentas, inserindo novos recursos às mesmas a fim de torná-las mais robustas e precisas (HOWE et al., 2008).

Os biocuradores podem ser considerados os catalogadores da era da Internet, pois transfor-

mam objetos inertes e não identificados em uma forma de representação poderosa passível de ser entendida e aprendida. Somente essa atividade já seria de grande contribuição para o mundo da ciência, mas a importância do trabalho desenvolvido pelos biocuradores pode ser ainda mais ampla (BOURNE; MCENTYRE, 2006). Extrair dados oriundos da literatura, organizando-os utilizando vocabulários pré-estabelecidos é uma das atividades mais importantes e que consomem mais tempo na biocuragem (HOWE et al., 2008). O papel do biocurador é dinâmico e evolui em paralelo com a bioinformática, fazendo a ponte entre o conhecimento científico gerado e a disponibilização desse conhecimento de forma ágil e estruturada (SALIMI; VITA, 2006). Atualmente, os biocuradores agilizam a inserção de novos dados nos bancos de dados, automatizam a curagem, padronizam os dados e auxiliam as comunidades de pesquisas interessadas em contribuir com o processo de anotação (HOWE et al., 2008).

Investimentos em tecnologias de alta-capacidade (*high-throughput technologies*), iniciadas nas análises de expressão gênica através dos microarranjos de DNA (sigla em inglês para *deoxyribonucleic acid*), são cada vez mais comuns nas pesquisas biológicas e biomédicas. Essas tecnologias avançaram, e continuam avançando, rapidamente nas áreas de genoma, transcriptoma, proteoma e metaboloma. Esse avanço contribui significativamente para o crescimento da quantidade de dados biológicos produzidos, criando uma constante demanda por dados confiáveis, consistentes e precisos (BURGE et al., 2012). Como os recursos bioinformáticos continuam a crescer, o papel do biocurador também está em constante desenvolvimento, acenando para um futuro promissor (SALIMI; VITA, 2006).

Informações curadas advindas da literatura servem como *padrão-ouro* nas análises computacionais, sendo utilizadas na avaliação de dados obtidos através de tecnologias de alta-capacidade e na aferição (*benchmarking*) de algoritmos de mineração de dados (MD). Atualmente, os limites dos domínios biológicos estão aumentando rapidamente. A fim de prover um entendimento rápido e confiável desses novos domínios, diferentes iniciativas estão sendo propostas, estabelecendo padrões e provendo um fácil e rápido acesso às informações curadas, sendo o *Gene Ontology* (GO), um bom exemplo dessa iniciativa (HOWE et al., 2008).

O GO é uma das principais iniciativas em bioinformática do mundo, cujo objetivo é padronizar a representação dos genes e seus produtos, provendo interconexões entre espécies e bancos de dados. O GO provê um vocabulário controlado de termos utilizado para descrever as características inerentes dos produtos gênicos. O número de termos, e consequentemente de produtos gênicos, no GO tem aumentado devido aos esforços de seus curadores, que utilizam anotações advindas da literatura e também advindas de inferências baseadas em funções ortólogas para a expansão da ontologia (GOCONSORTIUM, 2012).

Muitos trabalhos foram desenvolvidos envolvendo biocuragem e o GO (MARKOWITZ et al., 2009; BASU et al., 2006; TAO et al., 2007; SMID; DORSSERS, 2004; RICHARDS et al., 2010; DU et al., 2009; JOSLYN et al., 2004; YEH et al., 2003), estando o QuickGO e o CvManGO entre os principais.

O QuickGO é uma ferramenta web desenvolvida para navegação no GO e provê ao usuário acesso a todas as anotações disponibilizadas pelo grupo de anotadores do GO. Uma das vantagens do QuickGO é a rapidez e facilidade de uso e acesso às anotações, sendo uma ferramenta web de navegação do GO a exibir anotações, manuais e eletrônicas, para quase 190.000 espécies, provendo uma extensiva gama de filtros (HUNTLEY et al., 2009).

O CvManGO (sigla para *Computational versus Manual GO annotations*) traça relacionamentos entre anotações advindas da literatura e advindas de predições computacionais. Essa ferramenta avalia o relacionamento entre esses dois termos do GO (um vindo da literatura e o outro vindo de predições computacionais), procurando por discrepâncias, identificando genes que requerem atualizações em suas anotações. Essa ferramenta representou um passo importante na busca de mecanismos que auxiliem na revisão de literatura (PARK et al., 2012).

A contribuição fornecida pela comunidade científica, através da anotação manual, é altamente respeitada e é essencial para o entendimento dos complexos cenários biológicos. Essa contribuição é amplamente aceita e produz anotações precisas. Entretanto, o custo de se obter essas anotações manuais é muito alto, tanto na área financeira quanto no tempo gasto para obtê-las (SERINGHAUS; GERSTEIN, 2007).

No trabalho (Baumgartner Jr. et al., 2007), os autores propõem uma métrica para avaliar o processo de construção de conhecimento e a qualidade desse conhecimento gerado. Essa métrica sugere que, atualmente, o processo de curagem manual levará muito tempo para terminar as anotações apenas dos principais organismos modelos. Isso sugere que, se a produtividade dos curadores manuais se mantiver constante, somente a curagem manual nunca será suficiente para completar a anotação de todos os proteomas atualmente disponíveis.

Por sua vez, o estado atual da pesquisa em bioinformática tem gerado um paradoxo: por um lado, ela tem criado uma grande necessidade de anotações manuais (utilizadas como *padrão-ouro*) e esforços voltados para a análise desses dados; e por outro, ela tornou impossível, através de esforços puramente manuais, analisar todos os dados gerados. Dessa forma, a pesquisa em bioinformática desencadeou uma urgente demanda por ferramentas inteligentes, responsáveis por automatizar a transformação de dados brutos em conhecimento (BURGE et al., 2012).

Paralelo a esse desenvolvimento, outras áreas de pesquisa relacionadas ao Aprendizado

de Máquina (AM), Inteligência Artificial (IA), Estatística e Ciência da Computação tem direcionado seus esforços buscando a construção de sistemas computacionais com aprendizado sem-fim (CARLSON et al., 2010a). Esses sistemas computacionais inteligentes podem trabalhar como assistentes, auxiliando na resolução de tarefas de aprendizagem específicas. Podemos citar como exemplo o *Read the Web* [1].

Pesquisadores descobriram que a precisão de sistemas computacionais inteligentes (SCI) pode ser melhorada pelo acoplamento de múltiplos SCIs independentes. Dessa forma, ao invés de se utilizar um único SCI, acopla-se vários SCIs obtendo resultados mais precisos (BLUM; MITCHELL, 1998; KITTLER, 1998). Além disso, esses SCIs independentes podem, automaticamente, classificar dados que poderão ser utilizados futuramente em suas tarefas de aprendizado (CARLSON et al., 2010b). Dessa forma, esses sistemas computacionais de aprendizado sem-fim podem ser tratados de maneira similar aos SCIs combinados. A combinação de SCIs pode ser definida como um conjunto de sistemas computacionais no qual as predições geradas por cada SCI são combinadas (utilizando alguma estratégia pré-estabelecida), para classificar ou clusterizar novos registros (DIETTERICH, 1997).

Motivado pelo exposto, essa pesquisa tem por principal objetivo propor uma arquitetura computacional que utiliza princípios de aprendizado de máquina, princípios esses que nortearam a criação do aprendizado de máquina sem-fim, para auxiliar biocuradores do GO na tarefa de classificação de novos termos, tarefa essa de cunho totalmente manual. Essa arquitetura proposta utiliza AM semi-supervisionado (ZHU, 2005), combinando diferentes classificadores na rotulação de novas instâncias do GO. Essas novas instâncias se juntarão às instâncias anteriormente rotuladas e serão utilizadas no retreinamento da arquitetura computacional. Além do objetivo principal supracitado, essa pesquisa também tem por objetivo a construção de conhecimento de alto-nível na forma de simples regras SE-ENTÃO e árvores de decisão. Esse conhecimento gerado pode ser utilizado pelos biocuradores do GO na busca por padrões importantes presentes nos dados biológicos, revelando informações concisas e relevantes sobre o domínio da aplicação.

Essa Tese está estruturada da seguinte maneira:

- O Capítulo 2 traz informações pertinentes ao Aprendizado de Máquina Sem-Fim, seus princípios, além de trazer informações sobre o NELL (*Never-Ending Language Learner*), primeiro ambiente computacional inteligente a utilizar esse paradigma de aprendizado;
- O Capítulo 3 traz os detalhes em relação ao *Gene Ontology* (GO). Esses detalhes com-

[1] http://rtw.ml.cmu.edu/rtw/

preendem informações sobre a ontologia criada, tipos de arquivos gerados, histórico e conjuntos de dados utilizados nessa pesquisa;

- O Capítulo 4 traz detalhes da arquitetura proposta nessa pesquisa;
- O Capítulo 5 aborda informações pertinentes aos classificadores utilizados, detalhando os classificadores utilizados nessa pesquisa e aperfeiçoamentos que foram realizados ao Algoritmo Genético utilizado;
- O Capítulo 6 aborda os testes empíricos que foram realizados utilizando 7 (sete) configurações da arquitetura proposta. Além disso, esse capítulo traz comparativos entre os classificadores analisados, tempo de execução e informações sobre o conhecimento gerado. A fim de verificar a degradação ocorrida nas configurações propostas nessa pesquisa, esse capítulo traz uma comparação entre aprendizado semi-supervisionado e supervisionado, utilizando as configurações propostas;
- O Capítulo 7 traz as conclusões obtidas na pesquisa, além de discussões relacionadas a essas conclusões e os desdobramentos que essa pesquisa terá.

Capítulo 2

APRENDIZADO DE MÁQUINA SEM-FIM

Nesse capítulo será introduzido o Aprendizado de Máquina Sem-Fim, com seus principais objetivos e princípios. A introdução será descrita na Seção 2.1 e os objetivos e princípios serão descritos na Seção 2.2.

2.1 Introdução

O termo aprendizado de máquina sem-fim apareceu pela primeira vez no trabalho desenvolvido por Carlson e colaboradores (CARLSON et al., 2010a). Nesse trabalho, um sistema computacional inteligente foi construído, cujo principal objetivo estava relacionado à leitura e construção de conhecimento a partir de páginas advindas da web. Esse sistema roda 24 (vinte e quatro) horas por dia, 7 (sete) dias por semana, ininterruptamente, aumentando à cada página lida e processada, sua capacidade de aprendizado, isso é, o sistema utiliza o conhecimento gerado nas suas próximas iterações (CARLSON et al., 2010a).

Mais precisamente, o sistema é chamado NELL (sigla em inglês para *Never-Ending Language Learner*)[1] e ele é aplicado na tarefa de ”ler a web”continuadamente, extraindo fatos (conhecimento) de páginas web escritas em língua inglesa. Segue abaixo, mais detalhadamente, as duas tarefas executadas pelo NELL a cada dia (CARLSON et al., 2010a):

- Leitura: extrair informações a partir da web, informações essas que serão utilizadas no preenchimento de uma base de conhecimento;
- Aprendizado: utiliza as informações que foram depositadas na base de conhecimento, coletadas na tarefa de leitura, e todo o conhecimento construído nas etapas de aprendizado

[1] http://rtw.ml.cmu.edu

anteriores, para construir novos conhecimentos, buscando aumentar, a cada dia o nível de acurácia dos mesmos.

A tese que fundamentou o NELL está centrada na sua forma de aprendizado, construída considerando a grande redundância das informações presentes na web, onde os mesmos dados são apresentados várias vezes utilizando diversos formatos. Além desse ponto principal, pode-se citar outros importantes pontos aplicados ao NELL. São eles (CARLSON et al., 2010a):

- Aprendizado sem-fim;
- Busca pelo avanço do estado da arte na área de processamento de linguagem natural;
- Busca pelo desenvolvimento de uma base estruturada de conhecimento mundial.

O NELL lê e constrói conhecimento relacionado à categorias e relações. *Cidades*,*empresas* e *esportes* são exemplos de categorias e *PossuiEscritorioEm (cidade, localização)* é um exemplo de relação. Todas as categorias e relações presentes na base de conhecimento (ontologia) do NELL podem ser visualizadas no site do *NELL Knowledge Base Browser*[2]. O NELL pode aprender categorias e relações com base em diferentes *visões*, as quais tem como base (CARLSON et al., 2010a):

- Padrões textuais livres de formato obtidos através de sentenças na web;
- Dados semi-estruturados presentes em tabelas e listas de páginas web;
- Padrões morfológicos presentes nos sintagmas nominais;
- Cláusulas de Horn probabilísticas (habilita o NELL a fazer inferências entre novas relações e as relações já aprendidas).

Segue abaixo exemplos do conhecimento constuído pelo NELL a partir da web. Esses exemplos foram obtidos na página principal do *Read the Web*[3], com seus repectivos valores de confiança.

- *n7e* **is a** *dataset used within the scientific field of machine learning* - 94.5;
- *wis* **is a** *geopolitical entity* **that is a location** - 91.3;

[2] http://rtw.ml.cmu.edu/rtw/kbbrowser/
[3] http://rtw.ml.cmu.edu

- *real_time_impact* **is an** *event outcome* - 97.3;
- *jiang_rong* **is a** *writer* - 93.8;
- *allan_simpson coaches* **a sports team** - 93.8;
- *kobe_bryant* **is a person who** *has age* **30** - 100.0;
- **the companies** *cnn* **and** *network compete with* **eachother** - 96.9;
- *president_ronald_reagan* **is a politician who** *holds the office of president* - 93.8;
- *zuckerberg* **is a** *top member* **of** *facebook* - 100.0;
- *league_of_nations* **is** *headquartered in* **the country** *switzerland* - 100.0.

2.2 Objetivos e Princípios

Um dos principais objetivos do NELL é mostrar que o novo paradigma de aprendizado sem-fim é factível e pode trazer, aos sistemas de aprendizado de máquina, uma significativa melhora na capacidade de resolver problemas, quando comparados à abordagens tradicionais (CARLSON et al., 2010a).

Os princípios por trás do NELL exploram uma série de teorias e abordagens. Dentre todos esses princípios, pode-se destacar:

1. Aprendizado semi-supervisionado (ZHU, 2005);
2. Uso de acoplamento, através de *coupled functions* (CARLSON et al., 2010b);
3. Uso de amostras negativas;
4. Auto-supervisão (auto-correção ativa ou passiva);
5. Auto-reflexão.

Uma das maiores motivações para este trabalho de doutorado é dar início a investigações que possam viabilizar a construção de um Sistema de Aprendizado Sem-Fim (SASF) capaz de realizar a tarefa de um biocurador. Considerando-se, entretanto, a complexidade do tema, e considerando-se também, que o tempo necessário para se atingir o amadurecimento necessário para se ter um sistema de aprendizado sem-fim vai além do tempo disponível para a realização

de um trabalho de doutorado, nessa pesquisa, foram dados apenas os primeiros passos na busca do SASF biocurador. Para tanto, utilizou-se os princípios 1, 2, 3 e 4, ficando o princípio 5 para trabalhos futuros. Nesse sentido, no trabalho de doutorado aqui apresentado, não se pretende desenvolver mecanismos para que o sistema biocurador possa "refletir"sobre seu desempenho e, automaticamente, alterar seus algoritmos e métodos a fim de melhorar seu prórpio desempenho. Esta tarefa de auto-reflexão é importante para caracterizar um sistema de aprendizado sem-fim, mas não será abordada nessa pesquisa. Como forma de caracterizar melhor a ideia da autor-reflexão, pode-se utilizar como exemplo a autor-reflexão presente no NELL. Naquele sistema, a auto-reflexão está relacionada à construção de novos extratores e posterior uso desses extratores utilizados são todos oriundos do GO, não há a criação de novos acoplamentos como acontece no NELL, sejam oriundos do *feedback* dos biocuradores ou através do conhecimento gerado.

O aprendizado de máquina semi-supervisionado está localizado entre o aprendizado supervisionado e o não-supervisionado, utilizando na tarefa de aprendizado tanto dados rotulados, quanto dados não rotulados.

No aprendizado supervisionado, os dados de entrada são um conjunto de registros, onde cada registro é caracterizado por uma tupla *(x,y)*, onde x é um conjunto de atributos e o y o atributo alvo, designado como rótulo da classe. Dessa forma, o objetivo do aprendizado supervisionado é utilizar um modelo de classificação que mapeie cada conjunto de atributos x para um dos rótulos de classes y pré-determinados. No aprendizado não-supervisionado não há o atributo alvo (rótulo das classes). Dessa forma, cabe ao modelo de clusterização rolutar cada um dos registros de entrada, utilizando similaridades e diferenças entre os padrões existentes (TAN; STEINBACH; KUMAR, 2009).

Assim, no aprendizado semi-supervisionado tem-se um *dataset* $\mathrm{X} = \{x_1,x_2,x_3,\ldots,x_n\}$ onde o mesmo é um conjunto de n exemplos e pode ser dividido em dois conjuntos: o primeiro $X_k = \{x_1,\ldots,x_k\}$ com os rótulos $Y_k = \{y_1,\ldots,y_k\}$, e um segundo $X_n = \{x_{k+1},\ldots,x_n\}$, onde os rótulos são desconhecidos (CHAPELLE; SCHöLKOPF; ZIEN, 2006).

Dessa forma, utiliza-se todos os exemplos rotulados no treinamento de um classificador e após esse treinamento, o classificador é utilizado na rotulação dos demais exemplos (não rotulados). O aprendizado semi-supervisionado se mostra útil quando há uma grande quantidade de dados não rotulados e uma pequena quantidade de dados rotulados, obtendo assim, dados rotulados de maneira simples e rápida, diminuindo os gastos de tempo, esforço ou dinheiro, quando comparados com a rotulação manual (CHAPELLE; SCHöLKOPF; ZIEN, 2006).

Buscando simplicidade e rapidez nessa classificação, a área de aprendizado de máquina tem utilizado o aprendizado semi-supervisionado em várias áreas, tais como (CHAPELLE; SCHöLKOPF;

ZIEN, 2006):

- Reconhecimento de fala;
- Prospecção de estruturas 3D de proteínas;
- Classificação de páginas na web.

Como supracitado, dentre todos os princípios por trás do aprendizado de máquina sem-fim, o acoplamento foi um dos explorados nessa pesquisa. No NELL a ideia de acoplar a aprendizagem de múltiplas funções gerou um grande impacto positivo nos resultados obtidos. Nos princípios gerais do NELL, existem três tipos gerais de acoplamento. São eles (CARLSON et al., 2010b):

1. "Restrições de saída: Para duas funções $f_a : X \rightarrow Y_a$ e $f_b : X \rightarrow Y_b$, é possível criar uma restrição para os valores y_a e y_b para uma dada entrada x, e assim, pode-se exigir que f_a e f_b satisfaçam essa restrição. Por exemplo, se f_a e f_b são funções booleanas e $f_a(x) \rightarrow f_b(x)$, pode-se restringir $f_b(x)$ a ter o valor 1 sempre que $f_a(x) = 1$". No caso prático, o exemplo pode ser visto da seguinte maneira. Suponha as duas funções booleanas f_a (que toma como entrada um sintagma nominal x e retorna 1 para o caso de x ser o nome de um atleta) e f_b (que toma como entrada o mesmo sintagma nominal x e retorna 1 para o caso de x ser o nome de uma pessoa). Assim, a restrição $f_a(x) \rightarrow f_b(x)$ define que se x é identificado como atleta, x deve também ser uma pessoa.

2. "Composição de restrições: Para três funções $f_a : X_a \rightarrow Y_a$, $f_b : X_b \rightarrow Y_b$ e $f_c : X_a \times X_b \rightarrow Y_c$, pode-se ter uma restrição válida para y_a e y_b para um dado x_a e um dado x_b. Pode-se, por exemplo, exigir f_a e f_b para satisfazer essa restrição. Dessa forma, f_a e f_b poderiam ser usadas para "verificar o tipo" dos argumentos de f_c, de modo que $\forall x_a, \forall x_b, f_c(x_a, x_b) \rightarrow f_a(x_a) \wedge f_b(x_b)$". No caso prático da leitura da Web, essa restrição pode ser implementada tendo-se três funções f_a (que toma como entrada um sintagma nominal x_a e retorna 1 para o caso de x_a ser o nome de uma pessoa), f_b (que toma como entrada um sintagma nominal x_b e retorna 1 para o caso de x_b ser o nome de uma empresa) e f_c (que toma como entrada um sintagma nominal x_a e um sintagma nominal x_b, e retorna 1 para o caso de x_a ser o nome de uma pessoa que trabalha para a empresa x_b). Assim, a restrição $\forall x_a, \forall x_b, f_c(x_a, x_b) \rightarrow f_a(x_a) \wedge f_b(x_b)$"define que se x_a é identificado com uma pessoa que trabalha para a empresa x_b, necessariamente, x_a deve ser uma pessoa e x_b deve ser uma empresa.

3. "Restrições *multi-view-agreement*: Para uma função $f : X \rightarrow Y$, se X pode ser particionado em dois subconjuntos (ou visões) disjuntos, onde escreve-se $X = \langle X_1, X_2 \rangle$, e assume-se que ambos X_1 e X_2 podem ser utilizados como atributos para predizer Y, então pode-se aprender $f_1 : X_1 \rightarrow Y$ e $f_2 : X_2 \rightarrow Y$ e obrigá-los a concordar. Por exemplo, Y poderia ser um conjunto de possíveis categorias para uma dada página na web, X_1 poderia representar as palavras contidas nessa página e X_2 poderia representar as palavras contidas em *hyperlinks* apontando para essa página web (esse exemplo foi utilizado na configuração do *Co-Training* (BLUM; MITCHELL, 1998)).

Nessa pesquisa, as funções a serem utilizadas no aprendizado são classificadores e esses classificadores são acoplados a outros classificadores utilizando o acoplamento 1 (Restrições de saída). A forma como esses classificadores foram acoplados foi baseada em (VERMA; Hruschka Jr., 2012). Maiores informações sobre o acoplamento utilizado nessa pesquisa serão detalhados no Capítulo 4 dessa tese. Pode-se citar o trabalho de Amaral e Hruschka (AMARAL; Hruschka Jr., 2012) como exemplo da utilização de acoplamentos no contexto de aplicações no GO. Nessa pesquisa, os autores adaptaram o AG (chamado CEE) proposto por (AMARAL; Hruschka Jr., 2010), para que o mesmo utilizasse, em sua função de aptidão final, valores de aptidões obtidos em registros relacionados à classes e subclasses do GO. Os resultados obtidos mostraram que o uso de acoplamentos no CEE proporcionou um aumento na acurácia obtida para as três classes analisadas e uma diminuição no tamanho das regras SE-ENTÃO geradas.

Nessa tese, o uso de amostras negativas foi utilizado internamente em um dos classificadores utilizados, detalhados no Capítulo 4, Seção 4.1.

A auto-supervisão está relacionada ao uso de diversas "opiniões", oriundas de classificadores e/ou de fontes de dados distintas, o que permite que o sistema (autonomamente) confronte diferentes resultados de aprendizado e considere como corretos apenas os resultados nos quais as restrições impostas não são infringidas. Nessa pesquisa, a auto-supervisão é aplicada quando utiliza-se o acoplamento de classificadores, podendo esses serem aplicados à diferentes conjuntos de dados relacionados diretamente às classes ou subclasses.

O próximo capítulo, Capítulo 3, traz detalhes sobre o GO, abordando informações sobre a ontologia mantida e também com relação aos conjuntos de dados utilizados nessa pesquisa, além de outras informações.

Capítulo 3

GENE ONTOLOGY (GO)

Nesse capítulo será detalhada as caracterítiscas do GO, além da explanação sobre os *data-sets* utilizados nessa pesquisa.

3.1 Introdução

Atualmente, profissionais das áreas biológicas e biomédicas gastam uma grande quantidade de tempo e esforço na busca por informações existentes sobre cada área de pesquisa. Essa situação é ainda mais prejudicada devido às grandes variações existentes na terminologia empregada, utilizando em algumas situações o senso comum, situação essa, que corrobora para a ineficiência de buscas envolvendo pessoas e computadores (ASHBURNER et al., 2000).

O projeto GO é um esforço coletivo que foi criado devido à escassez de descrições consistentes, relacionadas à produtos gênicos, em diferentes bases de dados. Os colaboradores do GO desenvolveram três vocabulários estruturados e controlados (ontologia) que descrevem produtos gênicos em termos de seus processos biológicos, componentes celulares e funções moleculares. Essa ontologia tem por característica ser espécie-independente, isso é, os termos depositados não são agrupados levando em consideração a espécie a qual pertence (ASHBURNER et al., 2000).

O projeto GO tem por norte três atividades. São elas (ASHBURNER et al., 2000):

1. Desenvolvimento e manutenção de uma ontologia;
2. Anotação de produtos gênicos, o que implica a criação de associações entre a ontologia, genes e produtos gênicos, dentro das bases de dados que participam do GO;
3. Desenvolvimento de ferramentas que facilitam a criação, manutenção e uso dessa ontologia.

Como descrito anteriormente, o GO tem por foco o desenvolvimento e manutenção de uma ontologia, que tem por objetivo descrever os atributos de objetos biológicos. Esses objetos biológicos, também chamados de termos, podem ser agrupados em três domínios: processo biológico (BP, sigla em inglês para *biological process*), componente celular (CC, sigla em inglês para *cellular component*) e função molecular (MF, sigla em inglês para *molecular function*) (ASHBURNER et al., 2000).

Fazem parte do domínio BP, as operações ou conjuntos de eventos moleculares, com começo e fim definidos, importantes para o funcionamento das unidades vivas integradas, tais como: células, tecidos, órgãos e organismos. O domínio CC está relacionado às células, abordando suas partes constituintes ou seu ambiente extracelular. Por fim, o domínio MF aborda as atividades elementares de um produto gênico a nível molecular, tais como: ligação ou catálise (ASHBURNER et al., 2000).

```
format-version: 1.2
data-version: 2013-03-22
date: 21:03:2013 16:26

[Term]
id: GO:0000001
name: mitochondrion inheritance
namespace: biological_process
def: "The distribution of mitochondria, including the mitochondrial genome, into daughter cells after mitosis
or meiosis, mediated by interactions between mitochondria and the cytoskeleton." [GOC:mcc, PMID:10873824,
PMID:11389764]
synonym: "mitochondrial inheritance" EXACT []
is_a: GO:0048308 ! organelle inheritance
is_a: GO:0048311 ! mitochondrion distribution

. . .

[Term]
id: GO:0000028
name: ribosomal small subunit assembly
namespace: biological_process
def: "The aggregation, arrangement and bonding together of constituent RNAs and proteins to form the small
ribosomal subunit." [GOC:jl]
subset: gosubset_prok
synonym: "30S ribosomal subunit assembly" NARROW [GOC:mah]
synonym: "40S ribosomal subunit assembly" NARROW [GOC:mah]
is_a: GO:0022618 ! ribonucleoprotein complex assembly
relationship: part_of GO:0042255 ! ribosome assembly
relationship: part_of GO:0042274 ! ribosomal small subunit biogenesis

. . .
```

Figura 3.1: Fragmento da ontologia construída e mantida pelo GO

A Figura 3.1 mostra um fragmento da ontologia construída e mantida pelo GO. Essa ontologia é gerada diariamente e pode ser obtida através do portal do GO[1]. Ela é estruturada utilizando o formato OBO (*Open Biological and Biomedical Ontologies*) utilizado pela OBO-Edit, aplicação *open source* utilizada para a visualização e edição de ontologias. O formato OBO foi criado com o objetivo de fornecer às ontologias legibilidade humana, facilidade de análise, extensibilidade e mínima redundância (ASHBURNER et al., 2000).

[1] http://www.geneontology.org/GO.downloads.ontology.shtml

No fragmento apresentado, pode-se observar a presença de dois termos pertencentes ao GO (id: GO:0000001 e id: GO:0000028, escolhidos aleatoriamente). A quantidade de atributos que cada termo pode possuir é variável, sendo os atributos *id*, *name*, *namespace* e *def* os principais. O atributo *id* é o identificador do termo dentro da ontologia. No atributo *name* encontra-se o nome do termo, enquanto que o atributo *namespace* refere-se ao domínio ao qual aquele termo pertence (BP, CC ou MF). No atributo *def* é feita uma breve descrição do termo. Outros atributos podem aparecer, tais como: *synonym*, *is_a*, *part_of*, *regulates*, dentre outros (maiores detalhes sobre o formato OBO podem ser obtidos no portal do GO[2]) (ASHBURNER et al., 2000).

A ontologia do GO é estruturada na forma de um grafo, $G = (V,E)$, direcionado acíclico (DAG, sigla em inglês para *directed acyclic graph*). V é o conjunto de vértices do grafo, formado pelos termos presentes na ontologia e E é o conjunto de arestas do grafo, constituído pelos relacionamentos entre os termos da ontologia, podendo estar esses vértices, conectados a um ou mais vértices, desde que não formem ciclos.

As Figuras 3.2, 3.3 e 3.4 ilustram um fragmento do DAG para os domínios BP, CC e MF, respectivamente. Para o domínio BP, foram selecionados aleatoriamente quatro subdomínios: morte celular, regulação biológica, organização de componentes celulares ou biogênese e processo de desenvolvimento. Ligados aos subdomínio morte celular, estão ligados os subdomínios: citoxidade dependente de complemento, toxicidade mediada por leucócitos e morte de células de outros organismos. Ligados aos subdomínio morte de células de outros organismos, estão ligados mais dois subdomínios: citólise em outro organismo e morte de células em outros organismos envolvendo interação simbiótica. A mesma análise pode ser realizada para os fragmentos dos domínios CC e MF. É importante salientar que as Figuras 3.2, 3.3 e 3.4 são utilizadas para fins didáticos e representam uma pequena fração do DAG do GO.

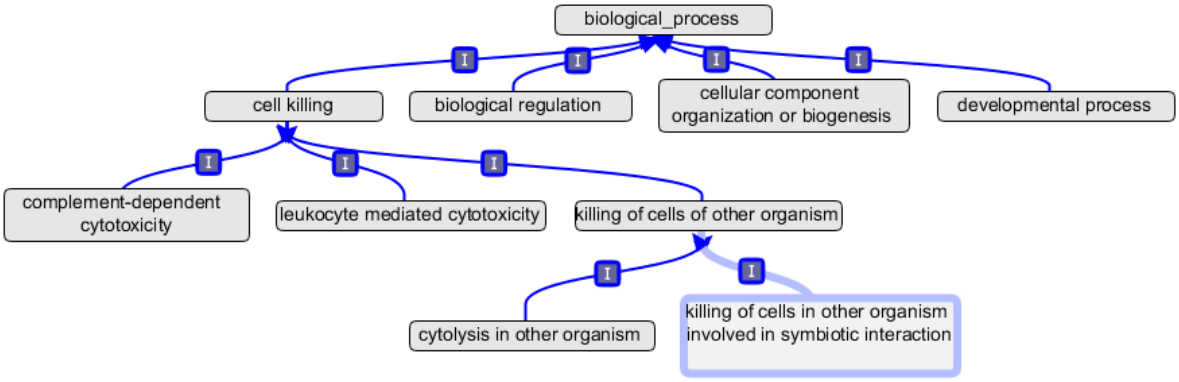

Figura 3.2: Fragmento do DAG para o domínio BP

[2]http://www.geneontology.org/GO.format.obo-1_2.shtml

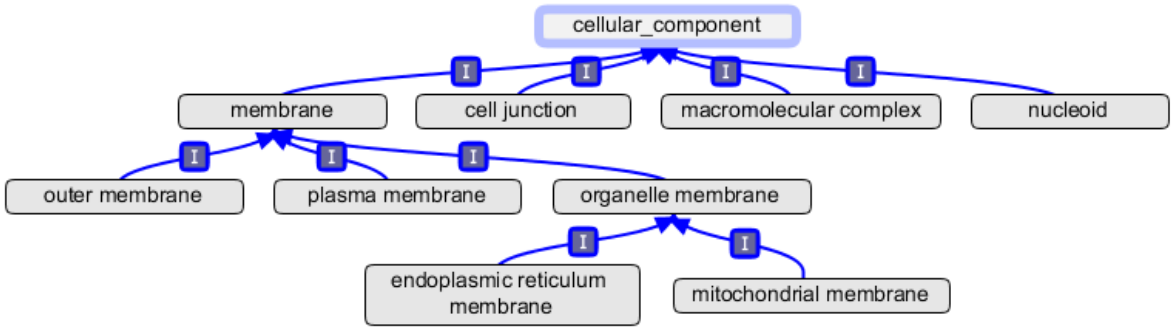

Figura 3.3: Fragmento do DAG para o domínio CC

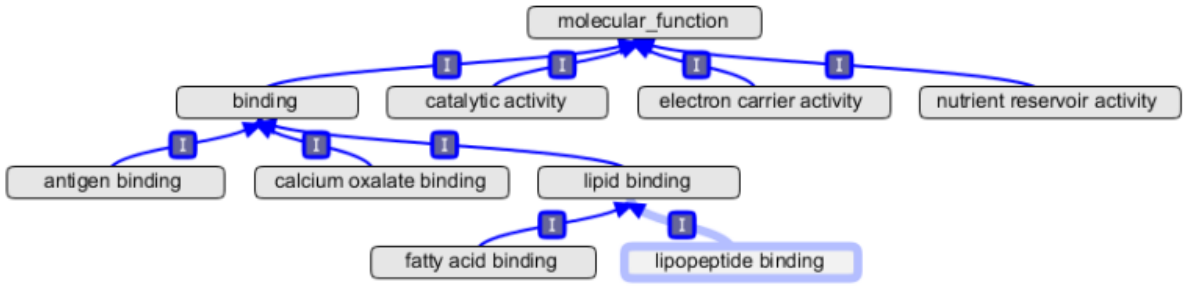

Figura 3.4: Fragmento do DAG para o domínio MF

Existem três diferentes tipos de arquivos que podem ser obtidos no portal do GO[3]. São eles: (ASHBURNER et al., 2000)

- *go-termdb* - os arquivos são gerados diariamente e contém somente os termos da ontologia e seus relacionamentos;
- *go-lite* (*assocdb* and *seqdb*) - os arquivos são gerados semanalmente e incluem: termos da ontologia, todos os produtos gênicos (com suas sequências) e seus relacionamentos, exceto os relacionamentos obtidos através de inferência eletrônica (IEA, sigla em inglês para *inferred from electronic annotation*);
- *go-full* - os arquivos são gerados uma vez ao mês e contém todos os dados, incluindo relacionamentos obtidos por IEA.

As bases de dados biológicas tiveram um crescimento considerável na última década, tanto em tamanho quanto em número (BURGE et al., 2012), e para o GO não foi diferente. A Tabela 3.1 mostra esse rápido crescimento no número de termos inseridos à ontologia do GO (base de dados *seqdb*, arquivos *go-lite*), de 2005 à 2012. Crescimentos significativos foram encontrados para o domínio CC e para os domínios BP + CC + MF, onde o número de termos quase dobrou.

[3]http://www.geneontology.org/GO.downloads.database.shtml

O maior crescimento ocorreu para o domínio BP, onde o número de termos quase triplicou. O número de relacionamentos entre os termos do GO também apresentou um alto crescimento, de aproximadamente 153%. De 2005 à 2012, o número de relacionamentos aumentou de 29.522 para 74.671. Se essa taxa de crescimento se mantiver constante, em 2020 o número de termos presentes no GO chegará próximo de 75.000 e os relacionamentos somarão 188.917, tornando a tarefa de curagem ainda mais complexa.

Tabela 3.1: Número de termos depositados no GO durante os anos 2005 e 2012

	Domínios			
Ano	**BP**	**CC**	**MF**	**BP + CC + MF**
2005	10.549	1.796	7.901	20.261
2006	12.905	1.976	8.050	22.952
2007	14.833	2.162	8.779	25.797
2008	16.286	2.370	9.214	27.898
2009	18.515	2.746	9.458	30.749
2010	20.622	2.915	9.733	33.309
2011	22.364	3.067	10.078	35.551
2012	25.072	3.251	10.422	38.791
Taxa de crescimento 2005-2012	**137,67 %**	**81,01%**	**31,90%**	**91,45%**

3.2 Datasets utilizados nessa pesquisa

Na busca por conjuntos de dados adequados aos propósitos dessa pesquisa, foram construídas quatro consultas SQL (sigla em inglês para *Structured Query Language*), utilizando dados advindos do banco de dados *seqdb*, arquivo *go-lite* do GO. Na construção das consultas SQL, que podem ser visualizadas no Apêndice A, foi utilizado o modelo de dados presente no Apêndice B.

Os dados oriundos dessas quatro consultas SQL foram divididos em dois conjuntos de dados: o primeiro, com 35 (trinta e cinco) atributos, está relacionado aos domínios BP, CC e MF da ontologia. O segundo grupo, com 34 atributos (foi retirado o atributo de código 10, fltDistDesvPadPais, ver Tabela 3.2), está relacionado aos subdomínios de BP, CC e MF, isso é, o atributo objetivo desse conjunto de dados são termos adjacentes aos domínios BP, CC ou MF dentro do DAG do GO. O atributo fltDistDesvPadPais foi retirado do segundo grupo pois retornou, para todos os registros do conjunto de dados, valor igual a 0 (zero). A partir desse ponto, serão tratadas as palavras domínio e subdomínio como classe e subclasse, respectivamente. Para fins de identificação o primeiro conjunto de dados será chamado de GOClasse e o

segundo, GOSubClasse.

A Tabela 3.2 traz detalhes de cada um dos 35 (trinta e cinco) atributos, onde a primeira coluna traz o código do atributo, a segunda traz o nome, a terceira o tipo do atributo e a quarta, traz uma breve descrição.

Na geração dos conjuntos de dados GOClasse e GOSubClasse foram utilizados dados gerados dia 16 de março de 2013, que podem ser obtidos em *http://archive.geneontology.org/lite/2013-03-16/*. A Tabela 3.3 mostra a distribuição dos registros para o GOClasse. Cada registro desse conjunto corresponde a um termo inserido à ontologia do GO por um biocurador. É importante salientar que devido à existência de relacionamentos entre várias tabelas do GO (como pode ser observado no Apêndice B), o número de registros utilizados nessa pesquisa é menor do que os mostrados na Tabela 3.1.

O conjunto de dados GOClasse foi dividido em 8 (oito) subconjuntos e a Tabela 3.4 traz essa distribuição. É importante salientar que a distribuição dos registros dentro dos subconjuntos foi feita de maneira estratificada, isso é, manteve-se em cada subconjunto, o mesmo número de registros para cada classe. Para a classe BP, a diferença foi de apenas um único registro, tendo os subconjuntos S_1 e S_2, 1.723 registros contra 1.724 dos demais. Para a classe CC as diferenças foram ainda menores, tendo 7 conjuntos com 239 registros e somente um (S_8) com 238. Para a classe MF, a diferença também não ultrapassou 1 registro, tendo os subconjuntos S_1, S_2 e S_3, 745 registros e os demais 746. Como esses registros serão utilizados dentro da arquitetura proposta será abordado no Capítulo 4, Seção 4.1.

A Tabela 3.5 traz importantes informações sobre o GOSubClasse. O conjunto de dados é constituído por 69.801 registros agrupados em 57 subclasses, sendo 23 relacionados à classe BP, 19 à CC e 15 à MF. Como pode-se perceber, os registros relacionados à classe BP estão em ampla superioridade numérica, respondendo por quase 80% dos registros. As Tabelas 3.6, 3.7 e 3.8 trazem outras importantes informações sobre o conjunto de dados GOSubClasse, complementando as informações apresentadas na Tabela 3.5.

A Tabela 3.6 traz informações sobre os registros relacionados à classe BP. Esses registros são agrupados em 23 subclasses, respondendo as subclasses *cellular process*, *single-organism process*, *biological regulation*, *regulation of biological process* e *metabolic process* por quase 60% dos registros.

Com relação aos registros relacionados à classe CC (Tabela 3.7), os mesmos estão distribuídos em 19 subclasses, respondendo as subclasses *cell*, *cell part*, *macromolecular complex*, *organelle* e *organelle part* por mais de 80% dos registros.

Tabela 3.2: Descrição dos atributos

Código	Atributo	Tipo	Descrição
01	intQtdeTermosAdjEhPai	Integer	Quantidade de filhos adjacentes
02	intQtdeTermosAdjEhFilho	Integer	Quantidade de pais adjacentes
03	intDistFilhoMax	Integer	Filho com maior distância
04	intQtdeFilhos	Integer	Quantidade de filhos
05	fltDistMediaFilhos	Double	Distância média de todos os filhos
06	fltDistDesvPadFilhos	Double	Desvio padrão da distância de todos dos filhos
07	intDistPaiMax	Integer	Maior distância de todos os pais
08	intQtdePais	Integer	Quantidade de pais
09	fltDistMediaPais	Double	Distância média de todos os pais
10	fltDistDesvPadPais	Double	Desvio padrão das distâncias de todos os pais
11	intNroSpecies	Integer	Número de espécies com este termo
12	intNroGeneProduct	Integer	Número de produtos gênicos
13	fltMediaProductCountSpecies	Double	Média de produtos gênicos por espécie
14	intQtdeSeqs	Integer	Número de aminoácidos na sequência
15	fltPorcAlanina	Double	Perc. do aminoácido (aa) alanina na sequência
16	dblPercArginina	Double	Perc. do aa arginina
17	dblPercAsparagina	Double	Perc. do aa asparagina
18	dblPercAcidoAspartico	Double	Perc. of aa ácido aspártico
19	dblPercCisteina	Double	Perc. do aa cisteína
20	dblPercAcidoGlutamico	Double	Perc. do aa ácido glutâmico
21	dblPercGlutamina	Double	Perc. do aa glutamina
22	dblPercGliycina	Double	Perc. do aa glicina
23	dblPercHistidina	Double	Perc. do aa histidina
24	dblPercIsoleucina	Double	Perc. do aa isoleucina
25	dblPercLeucina	Double	Perc. do aa leucina
26	dblPercLisina	Double	Perc. do aa lisina
27	dblPercMetionina	Double	Perc. do aa metionina
28	dblPercFenilalanina	Double	Perc. do aa fenilalanina
29	dblPercProlina	Double	Perc. do aa prolina
30	dblPercSerina	Double	Perc. do aa serina
31	dblPercTreonina	Double	Perc. do aa treonina
32	dblPercTriptofano	Double	Perc. do aa triptofano
33	dblPercTirosina	Double	Perc. do aa tirosina
34	dblPercValina	Double	Perc. do aa valina
35	strClasse	String	Classe

O menor número de subclasses está relacionado à classe MF (15 subclasses, Tabela 3.8),

Tabela 3.3: Distribuição dos registros do GOClasse

Classe	Número de registros	Percentagem dos registros
BP	13.790	63,65%
CC	1.911	8,82%
MF	5.965	27,53%
Total	**21.666**	**100%**

Tabela 3.4: Distribuição dos registros dentro dos 8 (oito) subconjuntos de dados para o GOClasse

Subconjuntos	BP	CC	MF	Total
S_1	1.723	239	745	2.707
S_2	1.723	239	745	2.707
S_3	1.724	239	745	2.708
S_4	1.724	239	746	2.709
S_5	1.724	239	746	2.709
S_6	1.724	239	746	2.709
S_7	1.724	239	746	2.709
S_8	1.724	238	746	2.708
Total	**13.790**	**1.911**	**5.965**	**21.666**

Tabela 3.5: Distribuição dos registros do GOSubClasse

	BP	CC	MF	Total
Número de subclasses	23	19	15	**57**
Percentagem das subclasses	40,35%	33,33%	26,32%	**100%**
Número de registros	55.799	7.577	6.425	**69.801**
Percentagem dos registros	79,94%	10,85%	9,21%	**100%**

onde mais da metade dos registros, mais precisamente 56,52%, está classificada como *catalytic activity*. A segunda subclasse com maior número de registros é *binding* com aproximadamente 19% e a terceira é *transporter activity* com 10,72%. Juntas essas três subclasses respondem por mais de 85% dos registros.

O próximo capítulo, Capítulo 4, traz detalhes da arquitetura proposta nessa pesquisa.

Tabela 3.6: Distribuição dos registros relacionados à BP no GOSubClasse

Subclasse	Número de registros	Percentagem dos registros
biological adhesion	112	0,20%
biological regulation	5.392	9,66%
cell killing	38	0,06%
cellular component organization or biogenesis	1.746	3,12%
cellular process	9.253	16,58%
developmental process	3.373	6,04%
establishment of localization	1.447	2,59%
growth	231	0,41%
immune system process	822	1,47%
localization	1.932	3,46%
locomotion	330	0,59%
metabolic process	4.466	8,00%
multi-organism process	473	0,84%
multicellular organismal process	473	7,09%
negative regulation of biological process	1.370	2,45%
positive regulation of biological process	1.515	2,71%
regulation of biological process	4.839	8,67%
reproduction	825	1,47%
reproduction II	677	1,21%
response to stimulus	2.698	4,83%
rhythmic process	60	0,10%
signaling	1.173	2,11%
single-organism process	9.066	16,34%
Total	**55.799**	**100,00%**

Tabela 3.7: Distribuição dos registros relacionados à CC no GOSubClasse

Subclasse	Número de registros	Percentagem dos registros
cell	1.647	21,73%
cell junction	29	0,38%
cell part	1.646	21,72%
extracellular matrix	47	0,62%
extracellular matrix part	41	0,54%
extracellular region	106	1,39%
extracellular region part	104	1,37%
macromolecular complex	1027	13,55%
membrane	429	5,66%
membrane part	348	4,69%
membrane-enclosed lumen	226	2,98%
nucleoid	6	0,07%
organelle	1.010	13,32%
organelle part	856	11,29%
symplast	1	0,01%
synapse	24	0,31%
synapse part	18	0,23%
virion	6	0,07%
virion part	6	0,07%
Total	**6.957**	**100,00%**

Tabela 3.8: Distribuição dos registros relacionados à MF no GOSubClasse

Subclasse	Número de registros	Percentagem dos registros
antioxidant activity	16	0,24%
binding	1.182	18,46%
catalytic activity	3.631	56,52%
channel regulator activity	10	0,15%
electron carrier activity	6	0,09%
enzyme regulator activity	141	2,19%
metallochaperone activity	3	0,04%
molecular transducer activity	316	4,91%
nucleic acid binding transcription factor activity	36	0,55%
protein binding transcription factor activity	41	0,63%
receptor activity	306	4,75%
receptor regulator activity	15	0,23%
structural molecule activity	29	0,45%
translation regulator activity	3	0,07%
transporter activity	690	10,72%
Total	**6.425**	**100,00%**

Capítulo 4

ARQUITETURA PROPOSTA

Nesse capítulo serão discutidos os detalhes da arquitetura proposta nessa pesquisa. Esses detalhes serão abordados na Seção 4.1.

4.1 Detalhamento da arquitetura

A arquitetura proposta é baseada em uma abordagem semi-supervisionada (ZHU, 2005), empregando princípios de acoplamento do aprendizado de máquina sem-fim, combinando classificadores independentes. A abordagem aqui apresentada foi baseada na abordagem proposta por (KROGH; VEDELSBY, 1995; WOLPERT, 1992; HUANG; SUEN, 1995), onde os classificadores envolvidos são agrupados em dois conjuntos: *classificadores de entrada* e *classificadores de saída*. A Figura 4.1 mostra a relação entre esses classificadores e demais detalhes da arquitetura proposta. Buscando detalhar o comportamento da arquitetura proposta mostrada na Figura 4.1, foi elaborado o Algoritmo 1. É importante ressaltar que o Algoritmo 1 é proposto para permitir que o processo tradicionalmente realizado por biocuradores humanos, seja aqui realizado automaticamente com a utilização de um conjunto de dados inicial e um conjunto de classificadores trabalhando de maneira acoplada e semi-supervisionada.

Como mostrado na Tabela 3.4, os 21.666 registros foram divididos em 8 subconjuntos ($S_{1..8}$), com aproximadamente 2.708 registros por subconjunto. Tal divisão é realizada para simular a chegada de novos dados não rotulados e que serão alvo do biocurador automático aqui proposto. O subconjunto S_1 foi classificado pelos biocuradores do GO e é utilizado no treinamento dos *classificadores de entrada* como dados rotulados. Com esses *classificadores de entrada* treinados, os mesmos são utilizados na classificação dos dados do subconjunto S_1 e as classificações obtidas são utilizadas no treinamento do *classificador de saída*. Ao final desse treinamento o *classificador de saída* é utilizado na rotulação do subconjunto S_2 (como mostrado

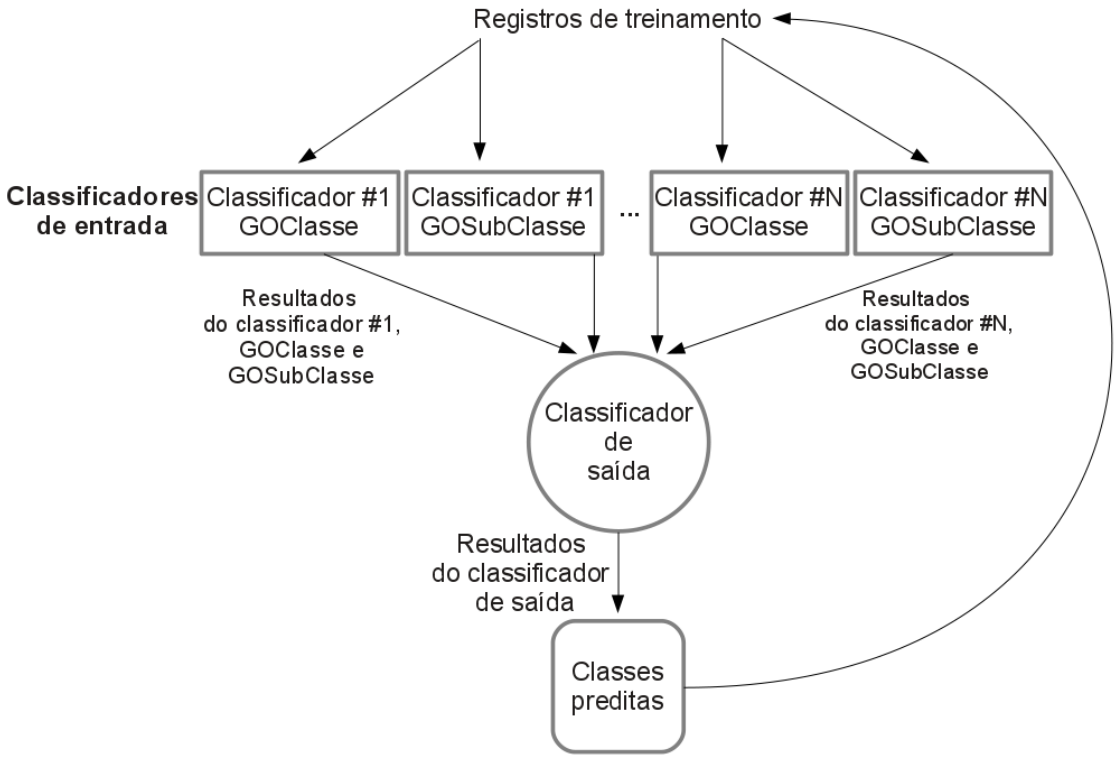

Figura 4.1: Arquitetura proposta

na Figura 4.2).

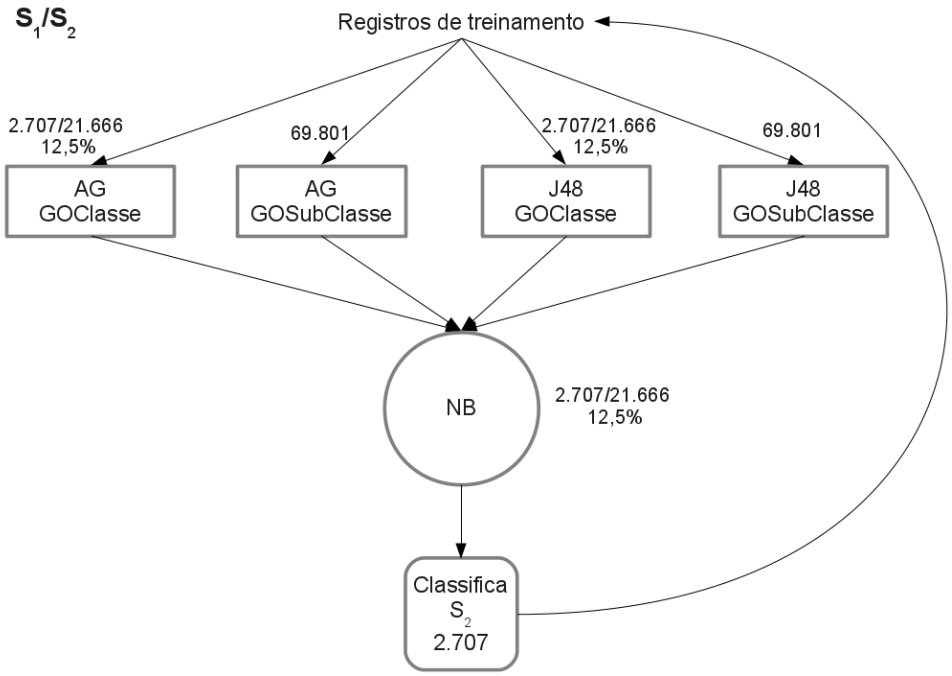

Figura 4.2: Etapa 1

Com os subconjuntos S_1 e S_2 rotulados, o primeiro pelos biocuradores do GO e o segundo pelo *classificador de saída*, os mesmos são utilizados em um novo treinamento dos *classificadores de entrada*. Com esses *classificadores de entrada* treinados novamente, os mesmos

Algorithm 1 ARQUITETURA PROPOSTA

Require: n = número de classificadores;
p = número de registros relacionados às classes;
GOSubClasse = base com dados relacionados às subclasses;
GOClasse[8][p] = subconjuntos com dados relacionados às classes;
classifClasse[n] = grupo de classificadores treinados na GOClasse;
classifSubClasse[n] = grupo de classificadores treinados na GOSubClasse;
CS = classificador de saída.
Ensure: GOClasse rotulada.

```
1: for i = 0 to n − 1 do
2:         treinar classifSubClasse[i] com GOSubClasse;
3: end for
4: for i = 0 to n − 1 do
5:         treinar classifClasse[i] com GOClasse[1];
6: end for
7: for j = 1 to GOClasse[1].length do
8:
9:         for k = 0 to n − 1 do
10:                classificar o registro GOClasse[1][j] utilizando classifClasse[k] e classifSubClasse[k];
11:        end for
12: end for
13: treinar o CS utilizando o output de classifClasse e classifSubClasse em GOClasse[1];
14: for j = 0 to GOClasse[2].length − 1 do
15:        rotular GOClasse[2][j] utilizando o CS;
16: end for
17: for i = 2 to GOClasse.length − 1 do
18:
19:        while houver alteração nos rótulos de GOClasse[i] do
20:
21:                for k = 0 to n − 1 do
22:                        treinar classifClasse[k] com GOClasse[1..i];
23:                end for
24:                for m = 1 to i do
25:
26:                        for j = 0 to GOClasse[m].length do
27:
28:                                for k = 0 to n − 1 do
29:                                        classificar o registro GOClasse[m][j] utilizando classifClasse[k] e classifSubClasse[k];
30:                                end for
31:                        end for
32:                end for
33:                treinar o CS utilizando o output de classifClass e classifSubClasse em GOClasse[1..i];
34:                if i < GOClasse.length then
35:
36:                        for j = 0 to GOClasse[i].length do
37:                                rotular GOClasse[i + 1][j] utilizando o CS;
38:                        end for
39:                end if
40:        end while
41: end for
```

são utilizados na classificação dos dados do subconjunto S_1 e S_2 e as classificações obtidas são utilizadas no treinamento do *classificador de saída*. Ao final desse treinamento o *classificador de saída* é utilizado na rotulação do subconjunto S_3 (como mostrado na Figura 4.3). E assim o processo continua até que todos os 8 subconjuntos estejam rotulados, como mostrado na Figura 4.4.

Note que esse processo simula a ideia de que originalmente existem dados já rotulados na base do GO (representados pelo conjunto S_1) e à medida que novos dados não rotulados vão chegando, o biocurador automático rotula os novos dados e passa a utilizá-los no conjunto de treinamento de seus classificadores. Na aplicação da rotulação final, foi utilizado uma es-

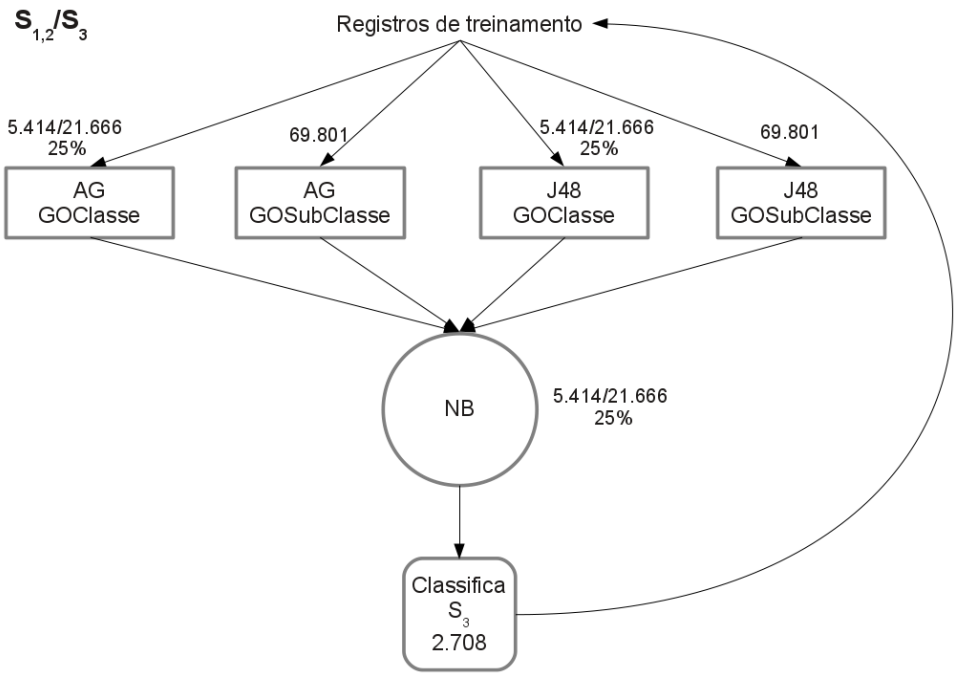

Figura 4.3: Etapa 2

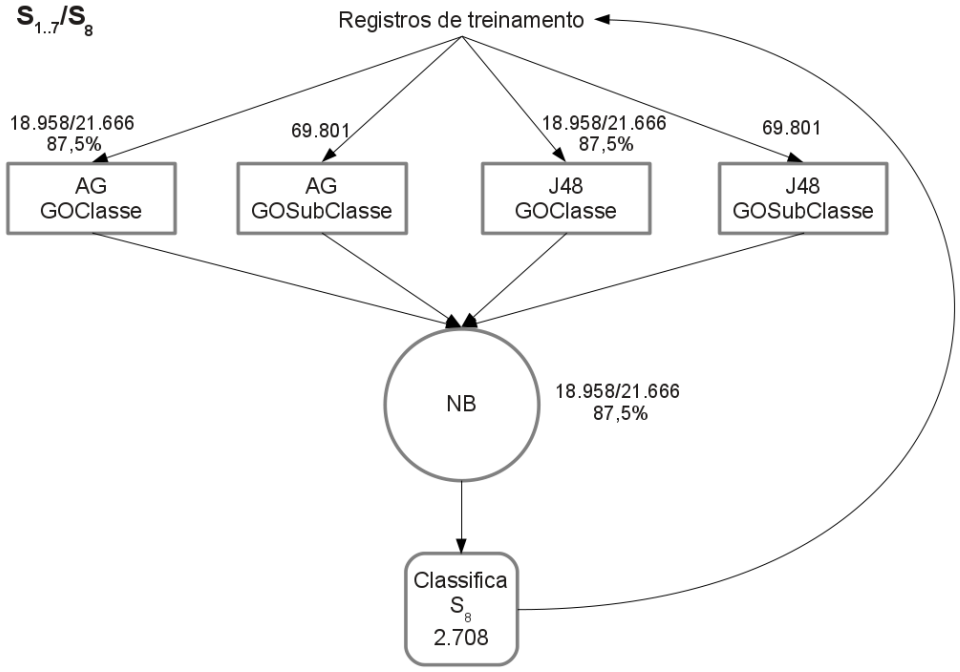

Figura 4.4: Etapa 7

tratégia de EM simplificada (DEMPSTER; LAIRD; RUBIN, 1977), isso é, passa-se à rotulação do subconjunto S_{n+1}, somente se não houver alterações na rotulação do subconjunto S_n.

Na composição dos *classificadores de entrada* foi utilizado o Algoritmo Genético (AG) (GOLDBERG, 1989) proposto por Amaral e Hruschka (AMARAL; Hruschka Jr., 2010) e uma árvore de decisão (QUINLAN, 1993). Com relação à árvore de decisão, foi utilizada uma implementação do C4.5 chamada J48, implementação essa presente no *software* Weka (FRANK et al., 2010). Como *classificador de saída* foi utilizado um classificador Naive Bayes (NB) (DUDA; HART, 1973). A Figura 4.6 ilustra uma das configurações utilizadas, chamada C_7, que utiliza 4 *classificadores de entrada*, dois AGs e dois J48. Cada conjunto AG/J48 é aplicado a registros relacionados à classes e subclasses do GO. Ainda com relação à configuração C_7, A Figura 4.5 traz o formato da base de treinamento fornecida ao classificador de saída. Como pode ser observado na figura, cada AG possui três atributos (um para cada uma das três classes, totalizando 6 atributos) e cada J48 somente 1 (totalizando 2 atributos).

```
@relation trainNB

@attribute GAC0 {true, false}
@attribute GAC1 {true, false}
@attribute GAC2 {true, false}
@attribute GASC0 {true, false}
@attribute GASC1 {true, false}
@attribute GASC2 {true, false}
@attribute J48C0 {BP, CC, MF}
@attribute J48SC0 {BP, CC, MF}
@attribute strClassePredita {null,BP, CC, MF}


@data
true,true,false, true,false,false, BP, BP, BP
false,true,false, false,true,true, CC, MF, CC
false,false,true, true,false,true, CC, MF, MF
...
```

Figura 4.5: Base de treinamento para o classificador de saída para a configuração C_7, gerada pelos classificadores de entrada

Os detalhes sobre os classificadores utilizados (AG, J48 e NB) serão descritos no Capítulo 5. Maiores detalhes sobre as configurações utilizadas serão apresentados no Capítulo 6, mais precisamente na Tabela 6.1.

Como uma das características desejadas para o biocurador automático aqui proposto é a produção de conhecimento de alto nível, que auxilie os curadores humanos a entender as decisões tomadas pelo curador automático, foi escolhido o AG proposto por Amaral e Hruschka (AMARAL; Hruschka Jr., 2010). Esse AG escolhido, gera regras SE-ENTÃO simples, com poucos atributos por classe, gerando uma única regra por classe, com bons valores de precisão. Além

disso, esse AG não se utiliza de regras *default*. Quando um método utiliza regras *default*, se nenhuma regra não disparar para um novo exemplo a ser classificado, esse novo exemplo será atribuído para a classe com maior número de registros (no conjunto de treinamento). Assim, em problemas com classes desbalanceadas, que é o caso do problema em questão, as regras *default* podem ser indesejáveis. Segue abaixo um exemplo de regra aplicada ao GOClasse que se utiliza da regra *default*.

SE (intQtdePais $\geq$ *11,5) E (fltDistDesvPadFilhos* $\geq$ *0,8011) ENTÃO strClasse* $=$ *MF SENÃO use a regra default*.

Como pode-se observar, a regra apresentada traz informações somente em relação à classe MF, isso é, nenhum conhecimento é gerado para as classes CC e BP. Além disso, nenhum novo exemplo será classificado como CC pois se a regra disparar será classificado como MF, senão será classificado como BP, classe com maior número de registros.

Como discutido no Capítulo 2, o AG supracitado trabalha internamente com amostras negativas, pois fornece ao *classificador de saída* classificações sobre todas as classes analisadas e não somente a classe predita, como acontece no J48, outro *classificador de entrada* utilizado nessa pesquisa.

O J48 foi escolhido por ser um tradicional método de mineração de dados, que consegue construir árvores de decisão com altos valores de precisão. O NB foi escolhido também por ser um classificador tradicional, simples, rápido e com altos valores de precisão.

O Capítulo 5, traz detalhes dos classificadores (AG, J48 e NB) utilizados nessa pesquisa.

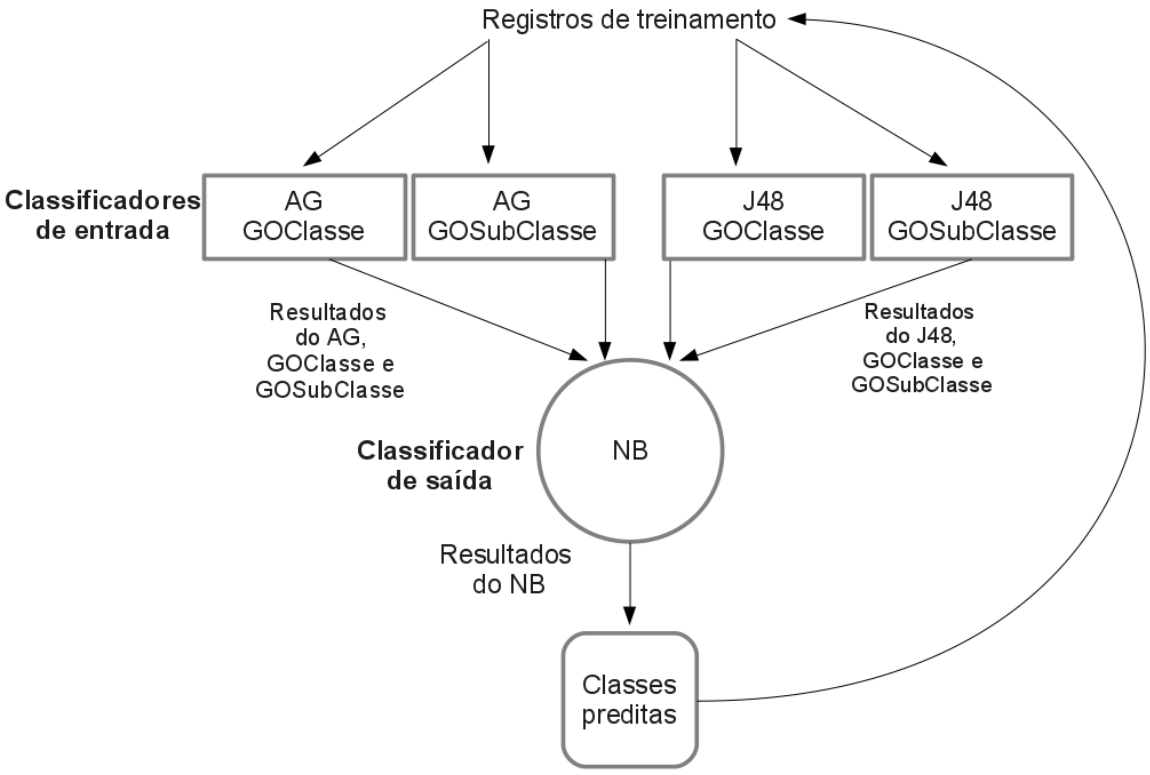

Figura 4.6: Configuração C_7

Capítulo 5

CLASSIFICADORES UTILIZADOS

Nesse capítulo serão descritos os detalhes sobre os métodos utilizados como *classificadores de entrada* e *classificadores de saída* na arquitetura proposta. A Seção 5.1 traz os detalhes sobre o AG utilizado nesse trabalho (subseções 5.1.1, 5.1.2 e 5.1.3). Além disso, ela também aborda os aperfeiçoamentos que foram desenvolvidos, tendo como ponto de partida o AG utilizado na arquitetura proposta, nesse trabalho de doutorado (Subseção 5.1.4). A Seção 5.2 traz os detalhes pertinentes ao J48 e ao NB.

5.1 Algoritmo Genético (AG)

Como supracitado, o AG utilizado nessa pesquisa foi proposto por Amaral e Hruschka (AMARAL; Hruschka Jr., 2010). No trabalho (AMARAL; Hruschka Jr., 2010), os autores propuseram um AG, chamado CEE (sigla em inglês para *Computational Evolutionary Environment*), cujo principal objetivo é a construção de conhecimento de alto nível na forma de regras SE-ENTÃO. Além desse objetivo, buscou-se também construir regras com um pequeno número de atributos (na parte SE da regra) e altos valores de acurácia, podendo as mesmas serem utilizadas como um sistema classificador. O CEE foi aplicado ao GO e como resultado os autores conseguiram construir regras simples, com baixo número de atributos (média de aproximadamente 3,5 atributos por classe) e com bons valores de sensibilidade e especificidade, com aptidão média de aproximadamente 69,67% por classe. Esses resultados foram obtidos em um *dataset* do GO.

Na construção do CEE, Amaral e Hruschka utilizaram o modelo proposto por Amaral e colaboradores (AMARAL et al., 2008), adequando-o para o GO. O modelo proposto por Amaral e colaboradores foi baseado no modelo proposto por Fidelis e colaboradores (FIDELIS; LOPES; FREITAS, 2000). Esse modelo utiliza regras SE-ENTÃO na representação do conhecimento, tendo por objetivo, classificar dados provenientes de registros clínicos de pacientes. Esses dados

clínicos abordam informações sobre idade, histórico familiar e sintomas do paciente. Mais de 94% dos atributos (32 de 34) estão relacionados aos sintomas do paciente e os mesmos foram discretizados da seguinte forma: 0 - ausente, 1 - ocorrência leve, 2 - ocorrência moderada e 3 - ocorrência severa. Esse conjunto de dados é constituído de 366 registros agrupados em 6 classes e o mesmo faz parte do *UCI Machine Learning Repository* (FRANK; ASUNCION, 2010).

Maiores informações sobre o AG utilizado nessa pesquisa serão abordados a seguir. Os detalhes sobre a representação do indivíduo será abordada na Subseção 5.1.1, função de avaliação na Subseção 5.1.2, operadores genéticos e parâmetros utilizados no AG na Subseção 5.1.3.

5.1.1 Representação do indivíduo

Para cada indivíduo, o *i*-ésimo gene (*i*=1...n), onde *n* é o número de genes do indivíduo, é subdividido em três subunidades: peso (W_i), operador (O_i) e valor (V_i), como mostrado na Figura 5.1. Cada gene corresponde a uma condição na parte *SE* (antecedente) da regra e o indivíduo, visto como um conjunto de genes, é todo o antecedente da regra.

O peso $W_i \in \mathbb{N}$; $1 \leq W_i \leq 10$; i = $\{1,2,3,\ldots n\}$. É importante salientar que o peso W_i é o responsável pela inserção ou exclusão de uma condição na parte *SE* da regra, isso é, caso o peso seja menor do que um valor limite *T* (*threshold*) o atributo referente a esse gene não fará parte da regra; caso contrário fará. O operador $O_i \in \{=,\neq,>,<,\geq,\leq\}$ e o operador $V_i \in \mathbb{R}$; k $\leq V_i \leq$ j, onde *k* e *j* são os menores e maiores valores, respectivamente, encontrados no conjunto de dados para o *i*-ésimo atributo. Essa estratégia foi adotada para aumentar o poder de convergência do AG, para que o mesmo não utilize valores em V_i muito discrepantes dos encontrados nos conjuntos de dados.

$Gene_1$			...	$Gene_n$		
W_1	O_1	V_1	...	W_n	O_n	V_n

Figura 5.1: Representação do indivíduo

A Figura 5.2 traz um exemplo de indivíduo com 4 genes ($n = 4$).

$Gene_1$			$Gene_2$			$Gene_3$			$Gene_4$		
9	$<$	1,7	6	$<$	8	10	$\geq$	3	4	$<$	2,1

Figura 5.2: Exemplo de indivíduo

O antecedente da regra equivalente a esse indivíduo (Figura 5.2) é dado por:

$$\text{SE } (Gene_1 < 1{,}7) \text{ E } (Gene_3 \geq 3)$$

Ou seja, somente $Gene_1$ e $Gene_3$ farão parte do antecedente da regra. $Gene_2$ e $Gene_4$ não farão parte do antecedente da regra, pois $W_2 <$ T e $W_4 <$ T (nesse exemplo, $T = 8$). O consequente não é representado explicitamente no indivíduo, isso é, a cada execução o AG busca regras de classificação para uma classe pré-especificada. Suponha que o indivíduo, exemplificado na Figura 5.2, represente uma regra para a classe C_1, dessa forma, a regra resultante é dada por:

$$\text{SE } (Gene_1 < 1{,}7) \text{ E } (Gene_3 \geq 3) \text{ ENTÃO } C_1$$

Na criação da população inicial são sorteados aleatoriamente os valores de W_i, O_i e V_i. As alterações de valores que podem ocorrer em W_i, O_i e V_i serão descritas na Subseção 5.1.3, quando for abordado o operador de mutação.

Nesse trabalho foram utilizados dois valores distintos para n. Para o AG que utilizou o GOClasse, $n = 34$ e para o AG que utilizou o GOSubClasse, $n = 33$. O atributo strClasse (Tabela 3.2) não é representado no indivíduo. O $Gene_i$ representa o i-ésimo atributo de GOClasse ou de GOSubClasse.

5.1.2 Função de avaliação (FA)

A FA utilizada foi baseada em (LOPES; COUTINHO; LIMA, 1997) e a cada execução o AG trabalha com um problema de classificação binária (duas classes), isso é, quando regras de uma dada classe C estão sendo mineradas, todas as outras classes são agrupadas em uma segunda classe $\neg C$.

Para o entendimento da FA alguns conceitos precisam ser elucidados. Ao se aplicar uma regra (um indivíduo da população) na classificação de um registro de GOClasse ou GOSubClasse, quatro diferentes resultados podem ser obtidos, dependendo da classe predita pela regra e da verdadeira classe do registro. São eles:

- *Verdadeiro Positivo* (*vp*) - A regra classifica o exemplo em uma determinada classe e o exemplo de fato pertence a essa classe;
- *Falso Positivo* (*fp*) - A regra classifica o exemplo em uma determinada classe, mas o exemplo não pertence a essa classe;

- *Verdadeiro Negativo* (*vn*) - A regra classifica o exemplo como não pertencente a uma determinada classe e o exemplo é de fato de outra classe;
- *Falso Negativo* (*fn*) - A regra classifica o exemplo como não pertencente a uma determinada classe, mas o exemplo pertence à classe em questão;

Como supracitado, uma regra pode classificar um determinado exemplo em *vp*, *fp*, *vn* ou *fn*. Ao final da classificação de todos os exemplos, cada regra possuirá um valor N_{vp} que totaliza o número de exemplos que foram classificados como *vp*. O raciocínio é análogo para N_{fp}, N_{vn} e N_{fn}. Assim, $N_{vp} + N_{fp} + N_{vn} + N_{fn} = m$, onde m = número de registros analisados. Utilizando N_{vp}, N_{fp}, N_{vn} e N_{fn}, calcula-se os valores de sensibilidade (*Se*) e especificidade (*Sp*) utilizando as Equações 5.1 e 5.2, respectivamente.

$$Se = \frac{N_{vp}}{(N_{vp} + N_{fn})} \tag{5.1}$$

$$Sp = \frac{N_{vn}}{(N_{vn} + N_{fp})} \tag{5.2}$$

O valor de aptidão de uma regra é obtido através da média aritmética simples entre *Se* e *Sp* e é definido pela Equação 5.3.

$$FA = (Se + Sp)/2 \tag{5.3}$$

O objetivo do AG utilizado é maximizar simultaneamente *Se* e *Sp* e, consequentemente, o valor da *FA*.

5.1.3 Operadores genéticos e parâmetros

Vários métodos e valores foram testados na busca pelos operadores genéticos e parâmetros que fornecessem regras com os mais altos valores de *Aptidão*. Na busca pela melhor configuração, cada possível combinação foi executada 35 vezes utilizando sementes randômicas distintas e os resultados obtidos foram comparados utilizando intervalo de confiança com $\alpha = 0.05$, isso é, com 95% de confiança. Devido ao significativo tempo de execução de cada combinação de parâmetros, escolheu-se 35 execuções, obtendo um bom grau de confiabilidade da estimativa. Segue abaixo os operadores genéticos e valores de parâmetros testados:

- Seleção de pais para *crossover*: roleta e torneio estocástico (*tour* = 3);

- *Crossover*: dois pontos de corte;
- Reinserção: elitismo e melhores pais e filhos;
- Tamanho da população: 100, 200 e 400;
- Número de gerações: 50, 100 e 200;
- Taxa de *crossover*: 100%;
- T (valor de *threshold* para o peso W_i): 5, 6, 7, 8 e 9;
- Taxa de mutação:
  - Aplicada ao indivíduo: 10, 20, 30 e 40;
  - Aplicada aos genes do indivíduo: 10, 20, 30 e 40.

Os melhores resultados foram obtidos utilizando torneio estocástico na seleção de pais para o *crossover*, melhores pais e filhos na reinserção, 100 indivíduos na população, evoluindo o ambiente por 50 gerações, $T = 8$ e com taxas de mutação de 30% e 30% aplicada ao indivíduo e aos genes do indivíduo respectivamente. O operador genético de mutação será aplicado aos novos indivíduos obtidos através do *crossover*, onde serão sorteados novos valores válidos para W_i, O_i e V_i, obedecendo restrições apresentadas na Subseção 5.1.1.

5.1.4 Aperfeiçoamentos realizados no CEE

Na busca por melhores taxas de aptidão, regras com menor número de atributos e convergências mais rápidas, foi proposto 4 (quatro) aperfeiçoamentos ao CEE (AMARAL; Hruschka Jr., 2010) supracitado. O primeiro, chamado GAFS (*Genetic Algorithm Feature Selection*), será descrito na Subsecção 5.1.4.1. O mesmo, é composto por um AG e um método de seleção de atributos, para ajudar o AG a selecionar atributos que sejam mais importantes para classificar registros de uma determinada classe. O segundo, chamado GANEL (*Genetic Algorithm Never-Ending Learning*), será apresentado na Subseção 5.1.4.2. Esse método utiliza de acoplamentos entre *datasets* na busca por melhores valores de aptidão. Na Subseção 5.1.4.3, será apresentado o NLCEE (sigla em inglês para *Non-Linear Computational Evolutionary Environment*), onde foi proferida mudanças no CEE, para que o mesmo possa ser utilizado em problemas onde o *dataset* apresenta caracaterísticas de não-linearidade. Na Subseção 5.1.4.4, será apresentado o operador *Transgenic*. Esse novo operador para AGs foi inspirado nos organismos geneticamente modificados (OGMs), onde importantes características são introduzidas artificialmente

dentro do genoma desses organismos. Nesse novo operador, um conjunto de importantes características são detectadas em alguns indivíduos da população. Essas importantes características são então introduzidas em *n* (passado como parâmetro de entrada) indivíduos da população, passando esses indíviduos a possuir essas importantes características.

5.1.4.1 Genetic Algorithm Feature Selection (GAFS)

Um dos problemas centrais em AM é a seleção de um subconjunto de atributos (FSS, sigla em inglês para *Feature Subset Selection*), onde a tarefa do algoritmo de aprendizado é selecionar um subconjunto de atributos, formado pelos atributos mais relevantes para a classificação (KOHAVI; JOHN, 1997), isso é, o algoritmo de aprendizado tem como objetivo escolher o melhor subconjunto de atributos, segundo algum critério em particular, que representa bem esses exemplos (GUYON; ELISSEEFF, 2003).

Buscando aumentar a precisão dos resultados e conferir um maior poder de convergência ao CEE, foi proposto o GAFS (AMARAL; Hruschka Jr., 2010). Dessa forma, alterações foram proferidas ao CEE criando um método híbrido que utiliza resultados oriundos de dois tradicionais métodos de seleção de atributos, *Chi-Squared* (CS) e *Gain Ratio* (GR) (presentes no ambiente Weka (FRANK et al., 2010)), na seleção dos genes que farão parte do antecedente das regras SE-ENTÃO.

Primeiramente foi aplicado esses dois métodos à base de dados GOClass, utilizando os valores *default* do Weka. As Tabelas 5.1 e 5.2 ilustram a saída dessa análise para o CS e GR respectivamente. Em uma primeira análise, foi selecionado somente os atributos que representaram mais de 1% do valor total (coluna "Valor absoluto"). Em ambos os métodos, foi selecionado 12 atributos.

Para que se pudesse utilizar os valores fornecidos pelos métodos CS e GR, os mesmos foram normalizados, trazendo seus valores para o intervalo de [1..3]. Para se chegar a esse intervalo supracitado, testou-se os seguintes valores máximos: 2, 3, 4, 5 e 7. Foi escolhido o valor 3, por ser o valor onde os melhores resultados foram encontrados. É vital informar que para chegar nesse valor máximo 3, o ambiente evolutivo foi executado 35 vezes aplicando sementes randômicas distintas e os resultados obtidos foram comparados utilizando teste de confiança com $\alpha = 0.05$. Como esses valores (2, 3, 4, 5 e 7) foram utilizados dentro do ambiente evolutivo será descrito abaixo. Com essa normalização, alguns atributos não obtiveram o valor mínimo (1%), sendo então excluídos do grupo. As Tabelas 5.3 e 5.4 ilustram os atributos selecionados e seus respectivos valores normalizados.

Tabela 5.1: Resultado obtido para o método *Chi-squared*

Ordem	Atributo	Valor absoluto	% do valor total
1	dblDistDesvPadPais	258.424.201	27%
2	dblDistMediaPais	189.581.850	19%
3	intQtdePais	158.285.635	16%
4	intDistPaiMax	78.844.090	8%
5	intQtdeTermosAdjEhFilho	75.762.607	8%
6	intDistDesvPadFilhos	40.322.459	4%
7	intDistMediaFilhos	39.420.291	4%
8	intDistFilhoMax	36.176.165	4%
9	intQtdeTermosAdjEhPai	35.687.420	4%
10	intQtdeFilhos	35.197.960	4%
11	fltMediaProductCountSpecies	11.247.578	1%
12	intNroGeneProduct	7.096.465	1%
Total		**966.046.721**	**100,00%**

Tabela 5.2: Resultado obtido para o método *GainRatio*

Ordem	Atributo	Valor absoluto	% do valor total
1	intQtdeTermosAdjEhFilho	0,130342	16,45%
2	dblDistDesvPadPais	0,097356	12,28%
3	intQtdePais	0,094959	11,98%
4	dblDistMediaPais	0,087987	11,10%
5	intDistPaiMax	0,06624	8,36%
6	dblDistDesvPadFilhos	0,063442	8,01%
7	intDistFilhoMax	0,06163	7,78%
8	intQtdeTermosAdjEhPai	0,061381	7,75%
9	intQtdeFilhos	0,060385	7,62%
10	dblDistMediaFilhos	0,050253	6,34%
11	dblMediaProductCountSpecies	0,009958	1,26%
12	intNroGeneProduct	0,008567	1,08%
Total		**0,7925**	**100,00%**

Tabela 5.3: Atributos selecionados e seus respectivos valores normalizados para o método *ChiS-quared*

Atributo	Valor normalizado
dblDistDesvPadPais	3
dblDistMediaPais	2
intQtdePais	2
intQtdeTermosAdjEhFilho	1
intDistPaiMax	1

Esses valores normalizados foram utilizados na construção da população inicial. Como descrito na subseção 5.1.1, o i-ésimo gene (i=1...N) é subdividido em três subunidades: peso (W_i), operador (O_i) e valor (Vi). A subunidade W_i é do tipo inteira e o seu valor está no intervalo

Tabela 5.4: Atributos selecionados e seus respectivos valores normalizados para o método *GainRatio*

Atributo	Valor normalizado
intQtdeTermosAdjEhFilho	3
dblDistDesvPadPais	2
intQtdePais	2
dblDistMediaPais	2
intDistPaiMax	2
dblDistDesvPadFilhos	1
intDistFilhoMax	1
intQtdeTermosAdjEhPai	1
intQtdeFilhos	1
dblDistMediaFilhos	1

[0..10], sendo essa subunidade responsável pela inserção ou exclusão de uma condição na parte antecedente da regra. Dessa forma, se o i-ésimo gene (que representa o i-ésimo atributo da base de dados) estiver na lista de atributos mostrada nas Tabelas 5.3 e 5.4, o valor de W_i, será calculado utilizando a equação 5.4, onde VN_i é o valor encontrado para o i-ésimo atributo na coluna ”Valor normalizado”das Tabelas 5.3 e 5.4.

$$Wi = Wi + VNi \tag{5.4}$$

Por exemplo, ao se analisar o gene que representa o atributo ”dblDistDesvPadPais”e o método selecionado é o *ChiSquared*, primeiramente, ocorre o sorteio de um valor para W_i, que nesse exemplo é igual a 6 (W_i = 6). Na Tabela 5.3, o valor normalizado para esse atributo é igual a 3 (VN_i = 3). Por fim, o novo valor para W_i será igual a 9 (W_i = 6 + 3), maior do que o valor sorteado anteriormente (6), aumentando a probabilidade do atributo ”dblDistDesvPadPais”fazer parte do antecedente da regra.

Além de aumentar a probabilidade de atributos que estão presentes nas Tabelas 5.3 e 5.4, há também a penalização dos atributos que não estão presentes nessas tabelas. A Equação 5.5 ilustra o cálculo de W_i para essa situação, onde o valor sorteado para W_i é decrementado em uma unidade. Isso é, se o valor sorteado para W_i é igual a 8 (W_i = 8), o valor final para W_i será igual a 7 (W_i = 7), diminuindo a probabilidade do i-ésimo atributo fazer parte do antecedente da regra SE-ENTÃO.

$$Wi = Wi - 1 \tag{5.5}$$

Uma extensão do GAFS, chamada HEE (*Hybrid Evolutionary Environment*) (SILVA; RIBEIRO; AMARAL, 2013), foi proposta por Silva e colaboradores. O HEE se mostrou muito eficiente na construção de regras SE-ENTÃO para o *dataset* NCI60 (ROSS et al., 2000,), tendo esse *dataset* mais de 1.000 atributos e somente 61 registros, obtendo valores de acurácia melhores que os obtidos para os métodos PART, J48, Random Forest, Naive Bayes e IBK. Maiores detalhes sobre o GAFS podem ser obtidos em (AMARAL; Hruschka Jr., 2010).

5.1.4.2 Genetic Algorithm Never-Ending Learning (GANEL)

Para se obter aprendizado contínuo e initerrupto, os sistemas de aprendizado de máquina podem ser desenvolvidos com base no paradigma de "aprendizado sem fim". Esse paradigma de aprendizado gera conhecimento continuadamente e incremental pois utiliza todo o conhecimento já obtido pelo processo de aprendizado na busca por resultados cada vez melhores. Dessa forma, O GANEL (AMARAL; Hruschka Jr., 2012) utiliza estratégias de acoplamento na busca de regras com um menor número de atributos e com melhores valores de precisão. É importante salientar que o GANEL incorpora princípios do NEL, tais como os acoplamentos, e não tem por objetivo prover aprendizado contínuo.

Enquanto o CEE foi evoluído utilizando os registros encontrados na base de dados GOClasse, o GANEL foi evoluído utilizando os registros presentes nas bases de dados GOClasse e GOSubClasse. Dessa forma, cada regra possui duas avaliações, chamadas aqui de ClassFit e SubClassFit. Utilizando essas duas avaliações, foi proposto dois acoplamentos chamados C_1 e C_2. No acoplamento C_1, a avaliação final de uma regra é calculada utilizando a Equação 5.6.

$$FinalFit = (ClassFit * W1) + (SubClassFit * W2) \tag{5.6}$$

onde W_1 e W_2 são valores pertencentes ao intervalo [0..1]. Por exemplo, se ClassFit = 0,62; SubClassFit = 0,59; W_1 = 0,6 e W_2 = 0,4; então FinalFit = 0,608. Para o acoplamento C_1, foram testados várias combinações, tais como: W_1 = 0,6 e W_2 = 0,4; W_1 = 0,7 e W_2 = 0,3; W_1 = 0,8 e W_2 = 0,2; e W_1 = 0,9 e W_2 = 0,1.

O acoplamento C_2 é utilizado quando a diferença encontrada entre ClassFit e SubClassFit é maior que um dado limiar (*threshold*) T_1. Por exemplo, se T_1 = 20% e ClassFit = 0,62; o acoplamento C_2 será aplicado se SubClassFit $<$ 0,496. Foi testado T_1 = 10%, T_1 = 15% e T_1 = 20%. O acoplamento C_2 é dividido em dois tipos: C_2P_1 e C_2P_2. No C_2P_1, o valor encontrado em ClassFit é decrementado de um valor fixo pré-estabelecido. A Equação 5.7 é utilizada no cálculo da avaliação final para o acoplamento C_2P_1.

$$FinalFit = ClassFit - P1 \tag{5.7}$$

onde P_1 é um valor pré-estabelecido. Por exemplo, se ClassFit = 0,60; P_1 = 0,05 e SubClassFit $<$ 0,48 (com T_1 = 20%) então FinalFit = 0,6 - 0,05 = 0,55. Em C_2P_2, o valor de ClassFit é decrementado de um valor variável, calculado utilizando os valores encontrados em ClassFit e SubClassFit. A equação 5.8 formaliza o cálculo da avaliação final para C_2P_2.

$$FinalFit = ClassFit - ((ClassFit - SubClassFit)/2) \tag{5.8}$$

Por exemplo, se ClassFit = 0,60; SubClassFit = 0,45 (ou menor que 0,48) e T_1 = 20% então FinalFit = 0,6 - ((0,6 - 0,45)/2) = 0,525.

O acoplamento C_1 é utilizado pelo GANEL na seleção de pais para o *crossover* e também no operador de reinserção, enquanto que o acoplamento C_2 foi utilizado apenas no operador de reinserção. Maiores detalhes sobre o GANEL podem ser obtidos em (AMARAL; Hruschka Jr., 2012).

5.1.4.3 Non-Linear Computational Evolutionary Environment (NLCEE)

O CEE é capaz de construir conhecimento de alto nível, na forma de regras do tipo IF-THEN, sendo apropriado para aplicações em bases de dados com distribuição linear dos dados. Devido a esse fato, ao aplicar esse método em bases de dados com características não-lineares, o método apresenta dificuldades de convergência além de não obter bons valores de aptidão. Partindo desta constatação, propomos uma extensão ao CEE adequando-o para minerar bases de dados que apresentem características de não-linearidade nos seus dados, chamado NLCEE (AMARAL; Hruschka Jr., 2011). A principal alteração efetuada foi na representação do indivíduo. A Figura 5.3 ilustra o esquema do indivíduo.

$Gene_n$					
WL_n	OL_n	VL_n	WR_n	OR_n	VR_n

Figura 5.3: Representação do indivíduo não-linear

Para cada individuo, o i-ésimo gene (i=1...N) é subdividido em seis subunidades: peso esquerda (WL_i), operador esquerda (OL_i), valor esquerda (VL_i), peso direita (WR_i), operador direita (OR_i) e valor direita (VR_i), como mostrado na Figura 5.3. Cada gene corresponde a uma condição na parte antecedente da regra (IF) e o individuo, visto como um conjunto de genes, é a parte antecedente da regra. As subunidades WL_i e WR_i são do tipo inteira e o seu valor está

compreendido entre [0..10]. É importante dizer que as subunidades WL_i e WR_i são responsáveis pela inserção ou exclusão de uma condição na parte antecedente da regra, isso é, caso o valor encontrado em uma destas subunidades seja menor do que um valor limite (*threshold*) o atributo referente a esse gene não fará parte da regra; caso contrário, o mesmo fará. A subunidades OL_i e OR_i podem receber os valores $<$ ou $\geq$ e a subunidades VL_i e VR_i variam entre o menor e o maior valor encontrado na base de dados para o í-ésimo atributo, podendo ser números inteiros ou ponto flutuantes.

As Figuras 5.4, 5.5, 5.6, 5.7, 5.8 e 5.9 ilustram as possíveis configurações (chamadas de S_1, S_2, S_3, S_4, S_5 e S_6, respectivamente) e suas regras resultantes encontradas para o indivíduo ilustrado na Figura 5.3. As Figuras 5.4 e 5.5 mostram o indivíduo e a regra resultante quando WL_1 é maior que 9 (limiar utilizado neste ambiente) e WR_1 é menor que 9. A parte esquerda do gene, formado pelas subunidades WL_1,OL_1 and VL_1, farão parte da regra pois $WL_1 > 9$. A parte direita do gene, constituído pelas subunidades WR_1, OR_1 and VR_1, não farão parte da regra pois $WR_1 < 9$.

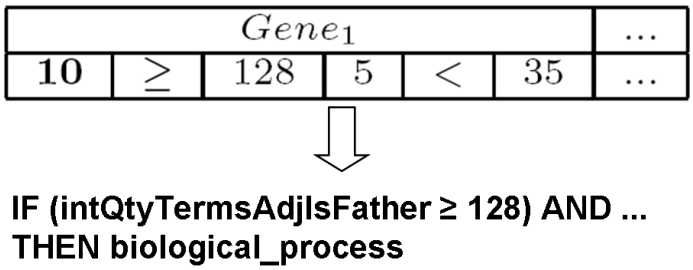

Figura 5.4: Configuração S_1 do indivíduo

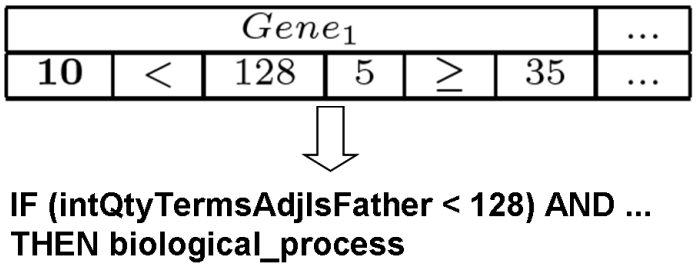

Figura 5.5: Configuração S_2 do indivíduo

As Figuras 5.6 e 5.7 mostram o indivíduo e a regra resultante quando WL_1 é menor que 9 e WR_1 é maior que 9. A parte esquerda do gene, formado pelas subunidades WL_1,OL_1 and VL_1, não farão parte da regra pois $WL_1 < 9$. A parte direita do gene, constituído pelas subunidades WR_1, OR_1 and VR_1, farão parte da regra pois $WR_1 > 9$.

Como mostrado nas Figuras 5.8 e 5.9, os indivíduos possuem as subunidades WL_1 e WR_1 maiores que 9. Nesse caso, tanto a parte esquerda quanto a direita do indivíduo farão parte

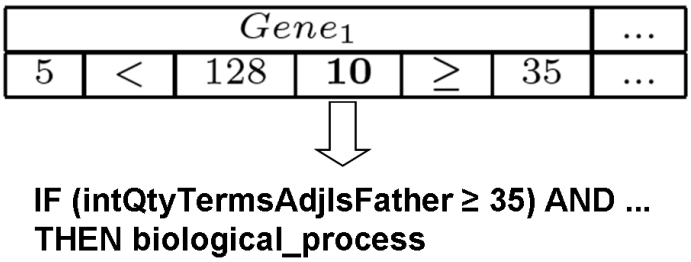

Figura 5.6: Configuração S_3 do indivíduo

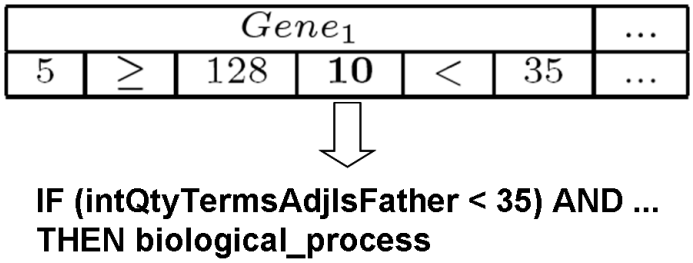

Figura 5.7: Configuração S_4 do indivíduo

da regra resultante. A Figura 5.8 mostra quando OL_1 é igual a $\geq$ e OR_1 é igual a $<$, levando a utilização do operador OR na regra resultante. A Figura 5.9, a regra resultante possui um intervalo de valores, pois a subunidade OL_1 do gene é igual a $<$ e a subunidade OR_1 é igual a $\geq$. Maiores detalhes sobre o NLCEE podem ser obtidos em (AMARAL; Hruschka Jr., 2011).

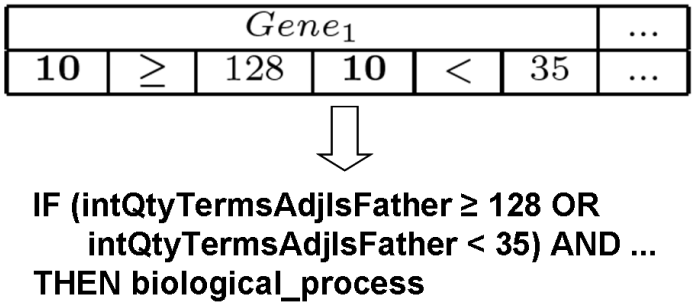

Figura 5.8: Configuração S_5 do indivíduo

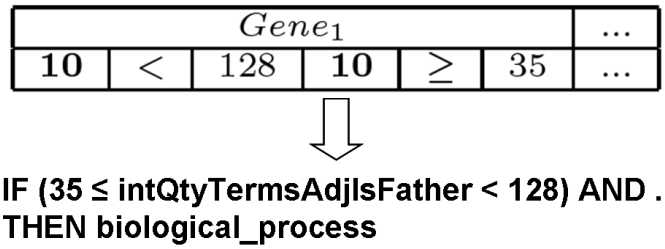

Figura 5.9: Configuração S_6 do indivíduo

5.1.4.4 Operador Transgenic

Em 1944, Oswald Avery identificou o DNA como a matéria-prima que forma os genes. A partir dessa descoberta, vários grupos de pesquisa se dedicaram a estudar o DNA, e sua composição foi rapidamente elucidada. O DNA é uma molécula constituída de açúcar, fosfato e quatro bases nitrogenadas denominadas adenina, citosina, guanina e timina. Posteriormente, os cientistas reconheceram essas quatro bases nitrogendas como sendo o alfabeto do código genético e passaram a denominá-las pelas iniciais A, C, G e T, também conhecidas como nucleotídeos (BORéM; SANTOS, 2004).

O ano de 1953 foi um marco para a genética, com a descoberta da estrutura helicoidal do DNA por dois cientistas da Universidade de *Cambridge*, Inglaterra: o americano James Watson e o inglês Francis Crick. Os trabalhos de ambos revolucionaram a genética e aceleraram as descobertas da estrutura do DNA. Eles demonstraram que a dupla hélice se constituía de duas fitas pareadas, cada uma com sua sequência de nucleotídeos, complementar a outra, isso é, na posição onde havia um A na primeira, aparecia um T na segunda, e onde havia um G aparecia um C e vice-versa (BORéM; SANTOS, 2004).

A partir deste marco, diveras áreas se desenvolveram significativamente ao longo dos anos e podemos citar a biotecnologia como sendo uma delas. A biotecnologia é a ciência que visa o desenvolvimento de produtos e serviços por processos biológicos, em geral utilizando a tecnologia do DNA recombinante, também conhecida como engenharia genética (COSTA; BORéM; BARBOSA, 2005). A biotecnologia aplica o conhecimento biológico de forma ativa às outras áreas da ciência, desenvolvendo tecnologias que implicam em melhorias na qualidade de vida do ser humano, otimizando de maneira racional a interação com a natureza, de modo que sejam supridas as necessidades de sobrevivência do homem. A área de saúde é a principal área de aplicação da biotecnologia, tendo hoje mais de 130 medicamentos e vacinas criadas (ALMEIDA; BORéM; FRANCO, 2004).

A engenharia genética ou também chamada de tecnologia do DNA recombinante pode ser definida como o conjunto de técnicas capazes de permitir a identificação, manipulação e multiplicação de genes dos organismos vivos. Mais precisamente, chamamos de transformação gênica a introdução de segmentos de DNA, chamados de transgenes, em um organismo. O objetivo final da transformação gênica é adicionar características novas ou diferentes ao indivíduo. Com a transformação gênica, o intercâmbio de genes, anteriormente limitado pelas barreiras sexuais da incompatibilidade entre espécies, deixou de existir, permitindo a troca de material gênico entre bactérias, animais e plantas (BORéM; SANTOS, 2004).

A transformação gênica tem gerado resultados de impacto na sociedade, como a produção de novos medicamentos e de alimentos com melhor qualidade nutricional. Um dos maiores sucessos comerciais da engenharia genética foi a síntese de insulina humana por bactérias em 1980. Outras espécies vegetais também foram transformadas, tais como: algodão, arroz, batata, canola e eucalipto. Em sua maioria, as características introduzidas nessas espécies foram tolerância a herbicidas, resistência a pragas e doenças, aumento da qualidade nutricional, dentre outras. A essas espécies supracitadas damos o nome de organismos geneticamente modificados (OGMs) (BORéM; SANTOS, 2004). Os OGMs são fabricados utilizando técnicas de engenharia genética, tais como: transgenia, mutagênese, desativação e/ou destruição de genes, dentre outras (ALMEIDA; BORéM; FRANCO, 2004).

Muitas idéias foram incorporadas aos AGs advindas da área de Genética, tais como: *crossover* haplóide, mutação, diploidia, deleção, dominância, translocação, diferenciação sexual (GOLDBERG, 1989) e introns (LEVENICK, 1991). Esses mecanismos trouxeram significantes características aos AGs, ampliando assim, a área de aplicação desse método (MITCHELL, 1999).

Dessa forma, foi proposto um operador, chamado *Transgenic* (AMARAL; Hruschka Jr., 2011, 2013), para AGs inspirado nos OGMs. O operador *Transgenic* manipula o material gênico dos indivíduos, introduzindo artificialmente importantes caracteríStcas a esses indivíduos. Esse método pode ser visto como uma estratégia de elitismo dirigida, isso é, focada em genes específicos. Dessa forma, o operador *Transgenic* identifica genes relevantes dentro de determinadas regras, seleciona e os aplica em indivíduos que possuem baixos valores de aptidão. Em outras palavras, se um determinado gene G é identificado por ter grande influência no valor de aptidão da regra, por possuir um determinado valor V, então, o operador *Transgenic* forçará alguns indivíduos da população, dotados de baixos valores de aptidão, a ter esse valor V no gene G na próxima geração.

A subunidade de peso W_i determina a inserção ou a exclusão do i-ésimo gene (atributo) na parte antecendente da regra, isso é, se $W_i > \mathrm{T}$ (onde T é um limiar que varia de [1..10]), então o gene (ou o atributo) fará parte do antecedente da regra. A cada geração do AG, o número de ocorrências dos genes (com $W_i > \mathrm{T}$) das melhores regras encontradas na população são incrementados, construindo um histórico. A Figura 5.10 ilustra a estrutura desse histórico.

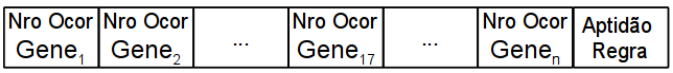

Figura 5.10: Estrutura do histórico

O histórico é composto por n posições, sendo n - *1* inteiras e uma ponto flutuante. As posições inteiras representam os n genes presentes no indivíduo. O campo ponto flutuante ar-

mazena a melhor avaliação presente na população. A construção do histórico se inicia com a população de regras. De posse dessa população, seleciona-se a(s) regra(s) que possue(m) o melhor valor de aptidão. Com esse subconjunto, analisa-se cada gene de cada regra, verificando se $W_i > \mathrm{T}$ (limiar pré-estabelecido). Se a subunidade peso for maior que um limiar pré-estabelecido, incremento a i-ésima posição do histórico. Essa análise é feita para todos os genes em todas as regras da população e ao final, tem-se uma lista de genes e seus respectivos números de ocorrências. A Figura 5.11 ilustra um exemplo deste histórico.

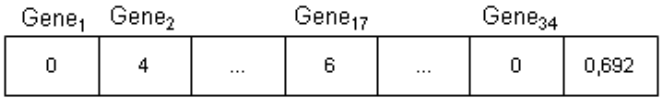

Figura 5.11: Exemplo do histórico

Nesse exemplo, tem-se $n = 34$ (pois existem 34 atributos no *dataset analisado*). O número de ocorrências para os genes 2 e 17 são maiores que 0 (zero) pois o gene 2 foi encontrado em 4 regras da população (que possuíam valores de aptidão igual a 0,692), e nessas regras o valor de W_2 era maior que o limiar, isso é $W_2 > 9$ (limiar utilizado neste trabalho). O gene 17 foi encontrado em 6 (seis) regras, que possuíam valores de aptidão igual a 0,692 e $W_{17} > 9$.

A partir desse histórico, é construída uma roleta onde cada um dos genes receberá uma fatia proporcional ao seu número de ocorrências. Isso é, se determinado gene tem um alto número de ocorrência então o mesmo terá uma porção maior da roleta. Dessa forma, há a garantia de que genes que apareceram em poucas regras podem vir a ser utilizados na transgenia, mas com probabilidade menor do que os mais significativos. Depois da construção da roleta, a mesma e executada n vezes, onde n é um parâmetro de entrada do operador *Transgenic* e limita o número de genes que participação da transgenia. Utilizando o exemplo de histórico apresentado na Figura 5.11, a roleta seria construída tendo o $Gene_2$ 4 unidades e o $Gene_{17}$ 6 unidades, tendo dessa forma, o $Gene_{17}$ uma maior probabilidade de ser selecionado para participar da transgenia. Esses n genes serão utilizados no módulos M_1, M_2, M_3 e M_4, descritos adiante.

Ainda com relação ao histórico, é importante salientar que se existirem na população outras regras com valores de aptidão iguais a 0,692 utilizando outros genes (com $W_i > 9$), o número de ocorrências para esses genes serão incrementados no histórico. Se uma nova regra aparecer na população com valor de aptidão maior do que 0,692, todos incrementos são zerados, e somente os genes com $W_i > \mathrm{T}$ (existentes nessa nova regra) serão incrementados.

A Figura 5.12 mostra a estrutura do operador *Transgenic*. O operador *Transgenic* é composto por 4 (quatro) módulos (M_1, M_2, M_3 e M_4) e utiliza uma estratégia similar à estratégia aplicada em (LATORRE et al., 2007). Esses módulos utilizam as informações existentes no histórico

para prover alterações nos indivíduos da população. Dessa forma, foram criadas 4 (quatro) cópias da população, uma para cada módulo, onde cada um desses módulos proverão alterações distintas nessas populações. Essas populações são então avaliadas e N indivíduos são selecionados, sendo N o tamanho da população. Esses N indivíduos irão compor a população transgênica.

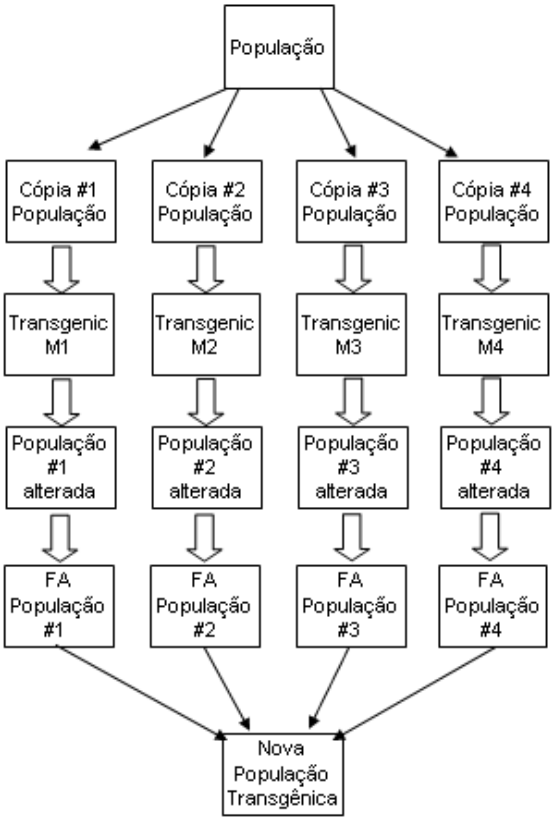

Figura 5.12: Estrutura do operador *Transgenic*

Os quatro módulos, apresentados na Figura 5.12, alteram o valor existente na subunidade W_i do gene, colocando-o maior que o limiar pré-estabelecido, nesse ambiente T = 9. Os quatro módulos recebem como entrada uma cópia da população e uma lista (construída utilizando a roleta, descrita anteriormente) com os genes que contribuem na obtenção daquele valor máximo de aptidão.

O módulo M_1 muda um único gene. Por exemplo, se o gene 17 está presente na lista de genes, o operador *Transgenic* altera o valor de W_{17}, para um valor maior que 9 (nesse caso, o único valor possível é 10). M_2, M_3 e M_4 fazem composições entre os genes presentes na lista 2-2, 3-3 e 4-4, respectivamente. Por exemplo, se a lista de genes do histórico for composta pelos genes de código 3, 11 e 19 e o módulo M_2 é utilizado, os valores de W_3, W_{11} e W_{19} serão alterados. Essa alteração ocorre com os genes dispostos dois a dois, isso é, irão ser alterados, em algumas regras, os valores de W_3 e W_{11}, em outras os valores de W_3 e W_{19} e em outras os valores de W_{11} e W_{19}, alterando-os para um valor acima do limiar.

Os módulos M_1, M_2, M_3 e M_4 trabalham em paralelo e ao final as quatro populações resultantes são compostas em uma única população os N melhores indivíduos (ordenados pela aptidão), onde N é o tamanho da população utilizada no AG. Esse operador não é aplicado em todos os indivíduos da população. Maiores detalhes sobre o operador *Transgenic* podem ser obtidos em (AMARAL; Hruschka Jr., 2011) e (AMARAL; Hruschka Jr., 2013).

5.2 Árvores de decisão e Naive Bayes

Uma árvore de decisão (QUINLAN, 1993) é uma estrutura de dados onde cada nó pode ser uma folha (indicando uma classe ou atributo objetivo) ou um nó de decisão. O nó de decisão especifica algum teste a ser realizado em um único atributo, possuindo um único ramo ou sub-árvore para cada um dos resultados encontrados. Uma árvore de decisão pode ser utilizada para classificar um exemplo iniciando pela raiz (*root*) da árvore e percorrendo-a até encontrar uma folha. Em cada nó não-folha, um teste é realizado utilizando o exemplo e dependendo do resultado, o fluxo é direcionado para uma das sub-árvores do nó não-folha. Quando esse processo finalmente (e inevitavelmente) é conduzido a uma folha, o exemplo é classificado com o rótulo dessa folha (QUINLAN, 1993). As árvores de decisão estão entre os métodos mais utilizados em aprendizado de máquina (PEDERSEN, 2001).

O classificador Naive Bayes, proposto por Duda e Hart (DUDA; HART, 1973), é o mais simples dos classificadores Bayesianos. Ele assume que todos os atributos envolvidos na classificação são condicionalmente independentes dado o contexto da classe. Os classificadores Naive Bayes são muito robustos quando aplicados em bases de dados com atributos independentes e sua predição final leva em conta vários atributos. Devido a independência condicional assumida, os parâmetros para cada atributo podem ser aprendidos separadamente, simplificando o processo de aprenzidado, especialmente quando o número de atributos é grande (MCCALLUM; NIGAM, 1998). O classificador Naive Bayes tem se mostrado eficiente em várias aplicações, tais como: classificação textual, diagnóstico médico e em sistemas de gestão de desempenho (DOMINGOS; PAZZANI, 1997; HELLERSTEIN; JAYRAM; RISH, 2000; MITCHELL, 1997).

Nessa pesquisa foi utilizado o *software* Weka em todos os experimentos envolvendo árvores de decisão e Naive Bayes. Foram utilizados os métodos J48, que é uma implementação do C4.5, e o Naive Bayes (NB). A Figura 5.13 e 5.14 mostram os parâmetros utilizados nos métodos J48 e NB, respectivamente. Nenhuma modificação nos parâmetros *default* foi realizada.

O próximo capítulo, Capítulo 6, traz informações sobre as configurações utilizadas na pesquisa, os resultados obtidos por essas configurações, analisando precisão, conhecimento gerado

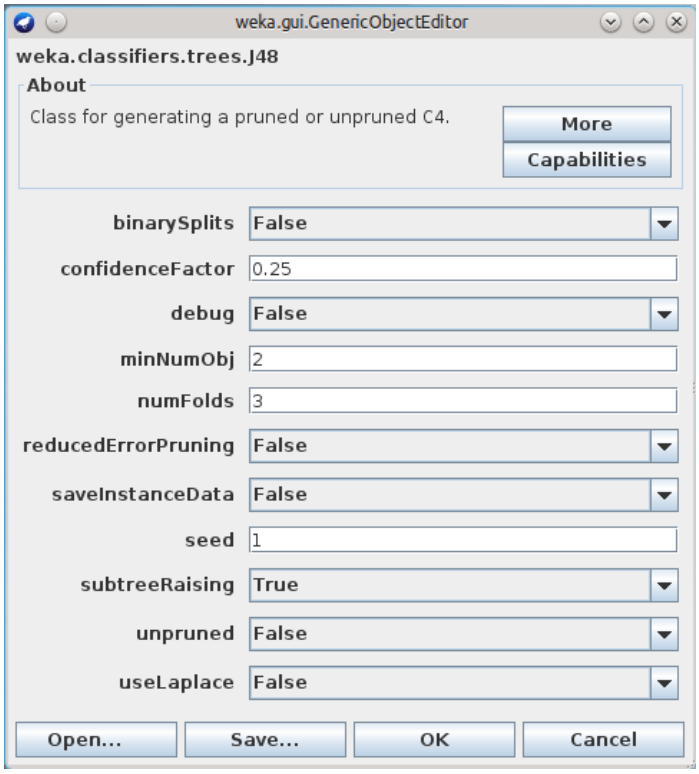

Figura 5.13: Parâmetros utilizados no J48

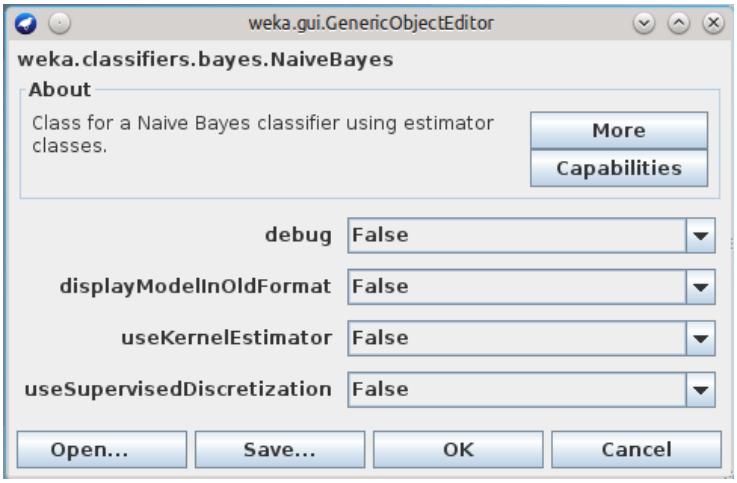

Figura 5.14: Parâmetros utilizados no NB

e tempo de execução.

Capítulo 6

RESULTADOS

Nesse capítulo será abordado as configurações utilizadas na pesquisa, os resultados obtidos pelas mesmas, analisando precisão, conhecimento gerado e tempo de execução.

6.1 Configurações utilizadas

Para melhor avaliar a arquitetura proposta no Capítulo 4, foram realizados estudos com avaliações empíricas com 7 (sete) configurações da arquitetura, CEE e J48. Essas configurações foram propostas a fim de exaurir todas as combinações possíveis ao se utilizar 2 (dois) classificadores de entrada e 1 (um) classificador de saída. A Tabela 6.1 ilustra as configurações utilizadas.

Tabela 6.1: Configurações utilizadas para a arquitetura proposta

Configuração	Descrição da configuração	Nro classif. de entrada
C_1	CEE + NB	1
C_2	CEE + NB + Subclasse	2
C_3	J48 + NB	1
C_4	J48 + NB + Subclasse	2
C_5	CEE + J48 + NB	2
C_6	CEE + J48 + NB + Subclasse (J48)	3
C_7	CEE + J48 + NB + Subclasse (CEE e J48)	4

Todas as configurações ($C_1, ..., C_7$) apresentadas na Tabela 6.1 utilizam o NB como *classificador de saída*. A configuração C_1 (Figura 6.1) utiliza o CEE como *classificador de entrada* e o conjunto de dados GOClasse. A configuração C_2 (Figura 6.2) expande a C_1, sendo o CEE aplicado aos conjuntos de dados GOClasse e GOSubClasse, isso é, um classificador CEE foi aplicado ao GOClasse e outro CEE ao GOSubClasse. Em C_3 (Figura 6.3), o J48 é utilizado como *classificador de entrada* e aplicado ao conjunto de dados GOClasse. O C_4 (Figura 6.4)

expande o C_3, aplicando o J48 ao GOClasse e GOSubClasse (um classificador J48 para cada conjunto de dados). Nas configurações $C_{5..7}$, o CEE e o J48 são utilizados como *classificadores de entrada*. Na configuração C_5 (Figura 6.5) é utilizado o conjunto de dados GOClasse e no C_6 (Figura 6.6) GOClasse e GOSubClasse, esse último, aplicado somente para o J48, totalizando três classificadores. Por fim, na configuração C_7 (Figura 6.7), CEE e J48 foram aplicados em GOClasse e GOSubClasse, totalizando 4 *classificadores de entrada*. Objetivou-se com as configurações $C_1, ..., C_7$ verificar a contribuição individual de cada método ou composição de métodos na classificação dos termos do GO, verificando ou não, entre outros pontos, a eficiência do uso do subconjunto de dados GOSubClasse nessa tarefa.

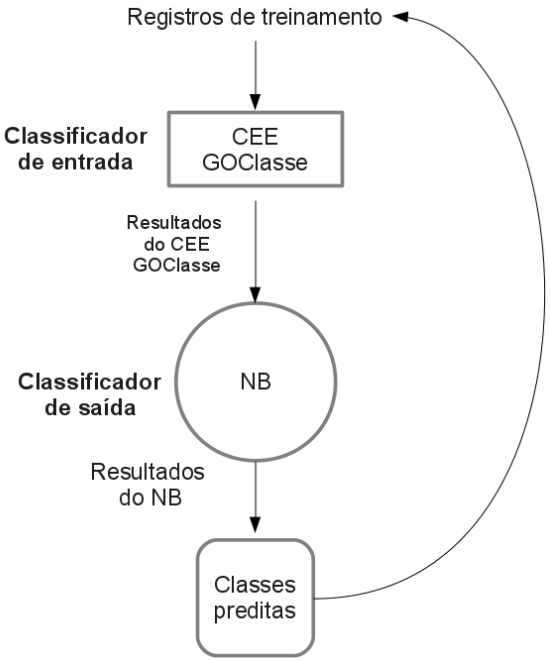

Figura 6.1: Configuração C_1

6.2 Resultados obtidos

A discussão dos resultados obtidos será dividida em duas vertentes. Primeiramente será discutida a precisão na classificação e posteriormente o conhecimento gerado pelo CEE, J48 e $C_1, ..., C_7$. Além disso, serão proferidas comparações entre precisão e conhecimento gerado, além de comparações entre a arquitetura proposta (semi-supervisionada) e suas variantes supervisionadas.

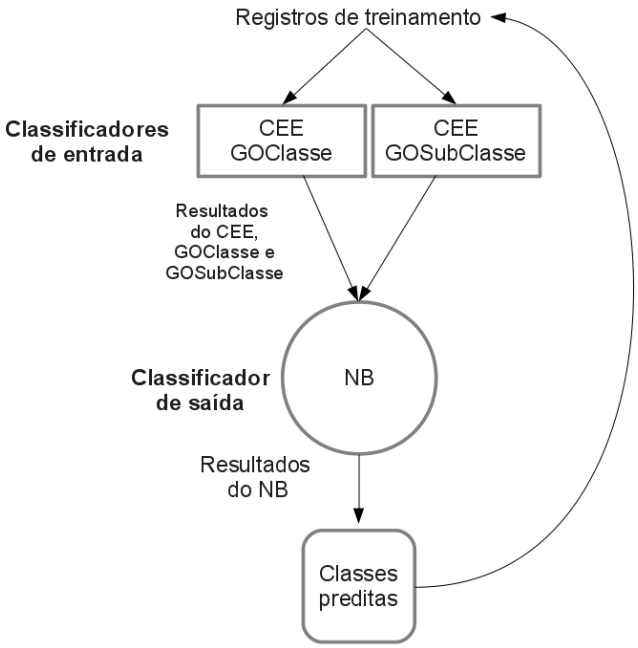

Figura 6.2: Configuração C_2

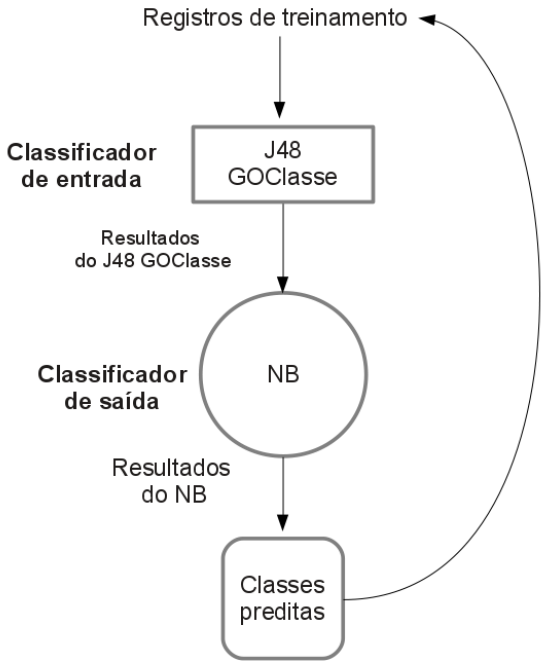

Figura 6.3: Configuração C_3

6.2.1 Precisão na classificação

A Tabela 6.2 traz os resultados de taxa média de classificação correta obtidos pelos métodos CEE e J48 aplicados individualmente, e também os obtidos pelas configurações $C_1, ..., C_7$, obtidos utilizando 10-*fold cross validation*. Esses resultados ilustram a percentagem de registros classificados corretamente nos subconjuntos $S_2, ..., S_8$. Por exemplo, se o método ou

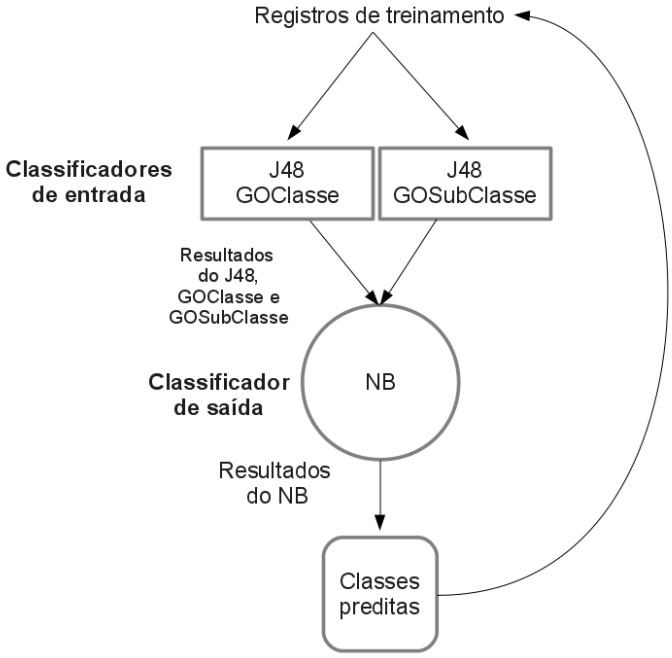

Figura 6.4: Configuração C_4

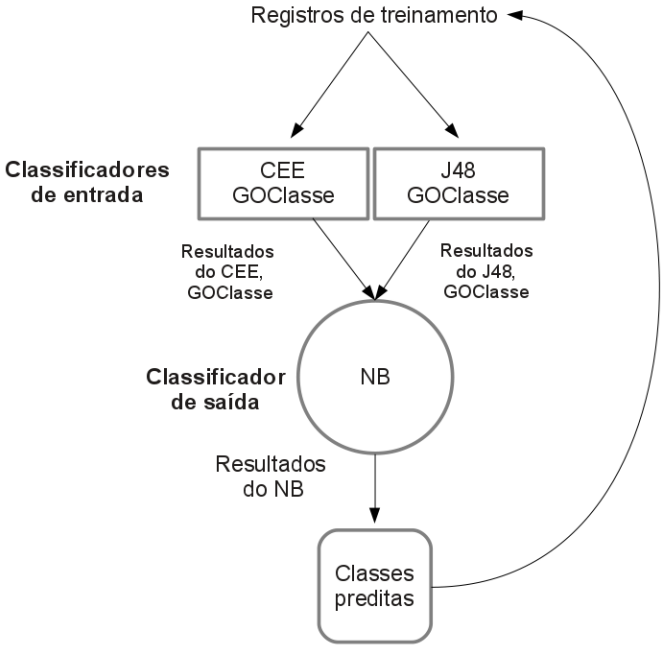

Figura 6.5: Configuração C_5

configuração obteve o resultado 80,56%, isso quer dizer que 80,56% dos registros dos subconjuntos $S_2, ..., S_8$ foram classificados corretamente. Na obtenção dos resultados para o método CEE e para as configurações $C_1, ..., C_7$ foi utilizado teste de confiança com $\alpha = 0.05$ (95% de confiança), executando o método ou configuração 35 vezes com sementes randômicas distintas. Para esse método ou configuração supracitada, o resultado está no formato: ValorMédio $\pm$

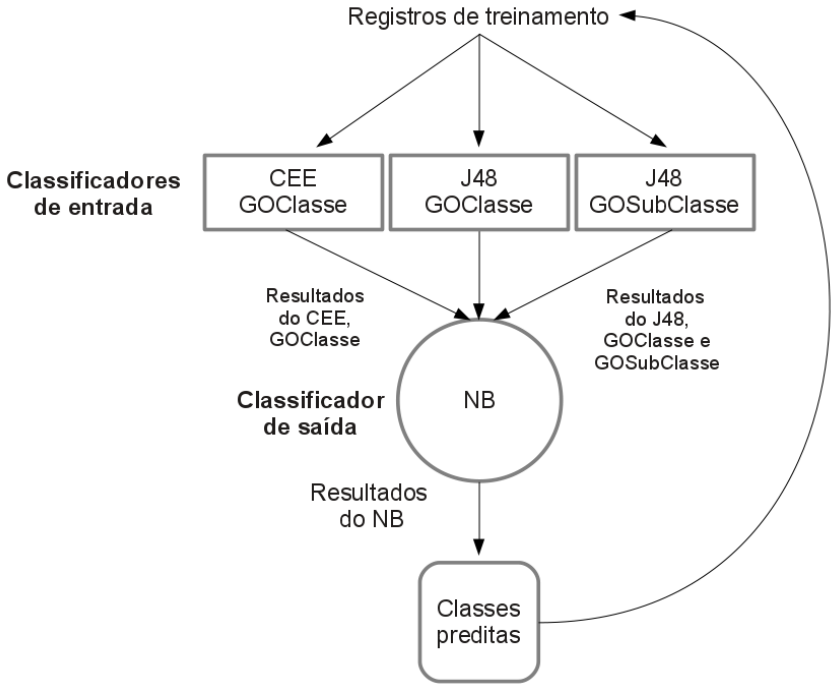

Figura 6.6: Configuração C_6

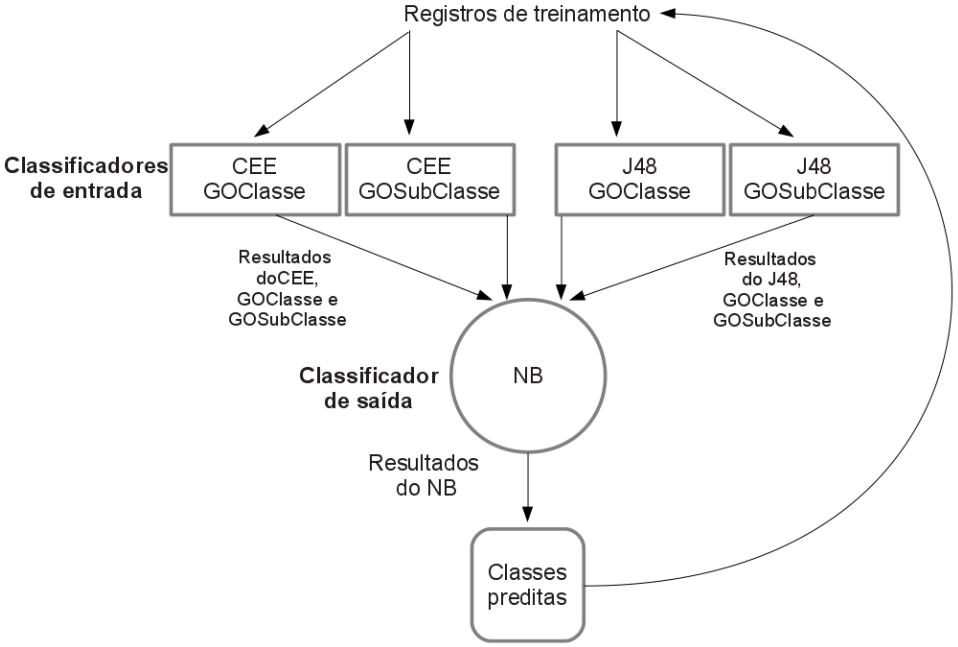

Figura 6.7: Configuração C_7

ValorConfiança[1]. Os valores obtidos nas 35 execuções podem ser visualizados no Apêndice C, Tabela C.1.

[1]O ValorConfiança é calculado utilizando o alfa, desvio padrão e o número de amostras analisadas. Por exemplo, se para uma deeterminada análise o valor médio é igual a 78,98, alfa = 0,05 e o ValorConfiança = 0,5, então posso encontrar valores variando entre 78,48 e 79,48 em 95% dos experimentos realizados

Tabela 6.2: Resultados encontrados para o CEE, J48 e $C_{1..7}$

Configuração/Método	Resultado
CEE	72,36 ± 0,83%
J48	83,21%
C_1	79,11 ± 0,31%
C_2	77,38 ± 0,25%
C_3	83,86%
C_4	93,98%
C_5	83,85 ± 0,09%
C_6	88,62 ± 0,32%
C_7	86,35 ± 0,35%

Considerando-se a precisão nas taxas de classificação obtidas, o melhor resultado foi obtido pela configuração C_4, obtendo aproximadamente 94% de classificação correta. Isso é, dos 18.959 registros encontrados nos subconjuntos $S_2, ..., S_8$, a configuração C_4 classificou corretamente 17.817, errando apenas 1.141 classificações. O segundo melhor resultado foi obtido por C_6, variando entre 88,3% e 88,94%. No pior e melhor caso, o C_6 errou 2.218 e 2.199 classificações respectivamente, errando 1.058 classificações a mais que o C_4 (no pior caso). O terceiro melhor resultado foi obtido por C_7, variando entre 86% e 86,7%, errando 2.654 e 2.521 classificações no pior e melhor caso respectivamente. Ainda analisando o pior caso, o C_7 classificou erroneamente 1.513 e 436 registros a mais quando comparados ao C_4 e C_6, respectivamente.

Para avaliar o ganho obtido com a adoção da arquitetura proposta, comparou-se os resultados obtidos pelo CEE com os obtidos em C_1 e C_2, bem como os resultados obtidos pelo J48 com os resultados obtidos em C_3 e C_4. Ao comparar C_1 com CEE, o ganho é de, no mínimo, 5,61 pontos percentuais, isso é, foram realizadas 1.063 classificações corretas a mais quando se utilizou C_1. Comparando C_2 com CEE, os resultados são, no mínimo, 3,94 pontos percentuais melhores, com 746 classificações corretas a mais com o uso de C_2. Ao comparar C_3 com J48, resultados 0,65 ponto percentual melhores foram encontrados, isso é, C_3 conseguiu classificar corretamente 123 registros a mais que o J48. Foram encontrados bons resultados na comparação entre C_4 e J48, sendo com a configuração C_4 o biocurador automático teve performance 10,77 pontos percentuais melhor do que com o uso do J48, classificando corretamente 2.041 registros a mais. Assim, a adoção da arquitetura proposta trouxe bons resultados quando comparados aos classificadores independentes, obtendo resultados (no mínimo) 5,61 e 10,77 pontos percentuais melhores que os encontrados para o CEE e J48, respectivamente.

Quando analisa-se o ganho obtido ao se utilizar a arquitetura proposta (utilizando somente o GOClasse) aplicado aos métodos CEE e J48, duas situações distintas ocorrem. Enquanto o ga-

nho com o CEE é relativamente bom, alcançando no mínimo 5,61 pontos percentuais, com o J48 é baixo, com somente 0,65 ponto percentual de ganho. Dessa forma, o uso da arquitetura proposta só traz ganhos significativos para o CEE. Por outro lado, ao analisar o ganho obtido pelo uso do GOSubClasse a situação se inverte. Enquanto o resultado do CEE deteriora, diminuindo 1,2 ponto percentual, o resultado do J48 aumenta em 10,12 pontos percentuais. Acredita-se que isso ocorra devido a simples estrutura de conhecimento utilizada pelo CEE e também pelo uso de sensibilidade e especificidade na avaliação das regras, quando aplicado à conjuntos de dados com desbalanceamento no número de registros entre as classes. Essa situação se repete quando se analisam as composições C_6 e C_7. Ao aplicar o GOSubClasse nos métodos CEE e J48 (C_7) os resultados encontrados decaem 1,6 ponto percentual.

Como mostrado na Tabela 6.2, não é sempre que a arquitetura proposta traz benefícios à acurácia. Ao comparar os resultados obtidos pelas configurações C_1 e C_2 aos obtidos pelo J48, percebe-se que os obtidos pelo J48 são, no mínimo, 3,79 e 5,58 pontos percentuais melhores, respectivamente.

6.2.2 Conhecimento gerado pelos classificadores

A Tabela 6.3 traz o tamanho da árvore de decisão gerada para o J48 e para as três configurações que obtiveram os melhores resultados (C_4, C_6 e C_7). Como pode-se perceber, não há grandes variações entre os resultados obtidos, exceto para C_4, que obteve um valor aproximadamente 62,2% maior que C_7 (segundo maior). Contrapondo esses resultados com os valores de acurácia apresentados anteriormente, a configuração C_4 obteve os melhores valores de acurácia, mas apresentou uma árvore de decisão bem maior que os demais.

Tabela 6.3: Tamanho da árvore de decisão gerada para o J48 e $C_{4,6,7}$

Método/Configuração	Tamanho da árvore
J48	975
C_4	1.747
C_6	1.019
C_7	1.077

A Tabela 6.4 ilustra o tamanho médio das regras (parte antecedente da regra) geradas pelo CEE e pelas configurações C_6 e C_7, nas 35 (trinta e cinco) execuções realizadas. Como pode-se perceber, não ocorreram grandes discrepâncias entre os valores analisados, variando de 5,4 a 5,51 (valor médio encontrado nas três classes). Além disso, pode-se perceber que as regras geradas possuem um pequeno número de atributos na parte antecedente, variando de 3,81 a 7,08.

Tabela 6.4: Tamanho médio das regras geradas para o CEE e $C_{6,7}$

Método/Conf.	Classe	Nro. médio de antecedentes	Valor médio BP-CC-MF
CEE	BP	3,81	5,4
	CC	6,93	
	MF	5,45	
C_6	BP	3,81	5,44
	CC	7,03	
	MF	5,48	
C_7	BP	4,08	5,51
	CC	7,08	
	MF	5,36	

6.2.3 Acurácia vs. conhecimento gerado

Como apresentado no Capítulo 1, o principal objetivo dessa pesquisa é propor uma arquitetura computacional que auxilie biocuradores do GO na tarefa de classificação de novos termos. Partindo desse presuposto, a Tabela 6.5 ilustra a comparação entre as configurações C_4 e C_6, trazendo valores de acurácia na classificação, tamanho da árvore de decisão gerada e tamanho médio das regras geradas (somente para C_6).

Tabela 6.5: C_4 vs. C_6: Acurácia e conhecimento gerado

	C_4	C_6
Acurácia	93,98	88,62 $\pm$ 0,32
Tamanho da árvore de decisão gerada	1.747	1.019
Tamanho médio das regras geradas	Não se aplica	5,44

Apesar da configuração C_4 apresentar valores mais altos de acurácia (no máximo 5,68 pontos percentuais melhor), a configuração C_6 fornece uma árvore de decisão 41,67% menor que a gerada em C_4. Além disso, a configuração C_6 permite a criação de regras SE-ENTÃO com médias de 5,44 antecedentes nas três classes analisadas. Na configuração C_6, pode-se ainda selecionar no conjunto de regras gerado, aquelas que obtiveram bons valores de aptidão e baixo número de antecedentes. Abaixo segue um conjunto de 3 (três) regras, mostradas na Tabela 6.6, com uma regra para cada classe analisada. A regra selecionada para BP possui somente 2 antecedentes, para MF, 3 e a regra selecionada para CC, possui 4 antecedentes, valores esses, menores do que os valores médios mostrados na Tabela 6.4. Deve-se considerar ainda, que o número de atributos utilizados nas três regras é pequeno, somente 4. Dessa forma, a configuração C_6 constitui uma ferramenta útil para o biocurador do GO, pois consegue agregar acurácia e conhecimento de alto-nível, através de uma árvore de decisão e três simples regras SE-ENTÃO.

Tabela 6.6: Regras geradas pelo C_6

SE (intDistPaiMax $<$ 7) E (intQtdePais $<$ 24) ENTÃO classe = BP
SE (intDistFilhoMax $>$ 2) E (intDistPaiMax $>$ 8) E (intQtdePais $\leq$ 24) E (dblPorcGlutamine $\geq$ 0,43) ENTÃO classe = CC
SE (intDistFilhoMax $\geq$ 1) E (intQtdePais $>$ 25) E (dblPorcGlutamine $\neq$ 0.81) ENTÃO classe = MF

6.2.4 Tempo de execução

A Tabela 6.7 traz o tempo de execução (em minutos) gasto por cada método ou configuração na rotulação de todas as instâncias dos conjuntos $S_2, ..., S_8$. Para as configurações C_1, C_2, C_5, C_6 e C_7 foi escolhida aleatoriamente uma semente randômica. Os dados presentes na Tabela 6.7 nos permite comparar, de forma superficial, o esforço computacional gasto por cada método ou configuração. Todos os experimentos foram executados em um laptop Dell Inspiron 14R 3350, com disco rígido de 750Gb, 6Gb de memória RAM, processador Intel Core i5, rodando Linux Ubuntu 12.04. O banco de dados utilizado foi o MySQL 5.5.31.

Tabela 6.7: Tempo de execução gasto pelos métodos/configurações

Método/Configuração	Tempo gasto (em minutos)
CEE	2,85
J48	0,93
C_1	3,67
C_2	9,02
C_3	2,75
C_4	2,90
C_5	7,92
C_6	17,32
C_7	21,67

Ao comparar o J48 com as configurações C_3 e C_4, pode-se perceber que o uso da arquitetura proposta elevou em 194,64% e 210,71% respectivamente o tempo de processamento. O uso do conjunto de dados GOSubClasse em C_4 não gerou um aumento significativo no tempo quando comparado a C_3, totalizando um aumento de 5,45%. Dessa forma, o uso da arquitetura proposta sem o auxílio do GOSubClasse (C_3) se mostrou uma opção não tão boa, pois o aumento na acurácia foi pequeno (0,6 ponto percentual) e o aumento de esforço computacional foi grande. Por outro lado, mesmo com um aumento no tempo de processamento em 210,71%, o C_4 conseguiu aumentar em 10,75 pontos percentuais a acurácia do J48.

Analisando o impacto da utilização da arquitetura proposta ao CEE, o resultado foi diferente do encontrado para o J48. O tempo gasto pela configuração C_1 foi 28,65% maior que o gasto

apresentado pelo CEE, longe dos 194,64% supracitados. Mas por outro lado, ao comparar o tempo gasto pelas configurações C_1 e C_2 o aumento é de aproximadamente 146%. Desta forma, a utilização do GOSubClasse não é aconselhada pois além de aumentar o tempo de execução, também traz uma queda na acurácia (como visto na Tabela 6.2). Contrapondo os resultados apresentados por C_5, C_6 e C_7, o aumento de tempo entre C_5 e C_6 foi de 118,73% e entre C_5 e C_7 foi de 166,10%. Entre C_6 e C_7 o aumento foi de 21,65%.

6.2.5 Aprendizado semi-supervisionado vs. supervisionado

Como citado no Capítulo 3, Tabela 3.4, a arquitetura proposta foi treinada utilizando-se os subconjuntos $S_1, ..., S_8$, com base no aprendizado semi-supervisionado. Dessa forma, somente S_1 foi rotulado pelos biocuradores do GO e todos os demais conjuntos, $S_2, ..., S_8$, foram rotulados pelo próprio sistema, implementado com base na arquitetura proposta. Uma das maiores motivações para a utilização do acoplamento de diferentes classificadores (como definido na arquitetura proposta) é a diminuição no número de classificações erradas geradas pelo desvio de conceito (Hruschka Jr.; DUARTE; NICOLETTI, 2013). Assim, a fim de avaliar o quanto os resultados obtidos pela arquitetura proposta deterioram ao longo do tempo, comparou-se as configurações C_4, C_6 e C_7 com uma abordagem supervisionada. Nessa abordagem supervisionada, as configurações C_4, C_6 e C_7 são treinadas nos subconjuntos $S_1, S_2, ..., S_{(N-1)}$ e testadas em S_N, utilizando somente dados rotulados pelos biocuradores do GO para o treinamento dos classificadores, ou seja, não há a semi-supervisão. A Tabela 6.8 traz os resultados obtidos para C_4, C_6 e C_7. Para C_6 e C_7, semi-supervisionado e supervisionado, cada configuração foi executada 35 vezes utilizando sementes randômicas distintas e os resultados obtidos foram comparados utilizando intervalo de confiança com $\alpha = 0.05$, isso é, com 95% de confiança.

Tabela 6.8: Semi-supervisionado vs. supervisionado

	C_4	C_4 S.	C_6	C_6 S.	C_7	C_7 S.
S_2	0,981	0,981	0,958 ± 0,031	0,958 ± 0,031	0,951 ± 0,033	0,951 ± 0,033
S_3	0,96887	0,98088	0,934 ± 0,006	0,957 ± 0,007	0,922 ± 0,005	0,948 ± 0,006
S_4	0,95281	0,98003	0,920 ± 0,011	0,955 ± 0,005	0,905 ± 0,007	0,947 ± 0,002
S_5	0,94778	0,97953	0,910 ± 0,013	0,952 ± 0,008	0,891 ± 0,014	0,945 ± 0,012
S_6	0,94497	0,97946	0,898 ± 0,010	0,949 ± 0,003	0,881 ± 0,020	0,942 ± 0,015
S_7	0,94176	0,97835	0,891 ± 0,005	0,940 ± 0,012	0,872 ± 0,018	0,939 ± 0,012
S_8	0,93994	0,97793	0,886 ± 0,003	0,938 ± 0,005	0,863 ± 0,003	0,929 ± 0,006

Como pode-se perceber, os resultados obtidos pela arquitetura proposta são próximos aos obtidos pelos ambientes supervisionados, ou seja, os resultados obtidos na abordagem semi-

supervisionada estão próximos dos resultados ótimos (onde os rótulos teriam sido fornecidos por especialistas humanos). Na comparação realizada paras as configurações C_4, C_6 e C_7 as diferenças foram de apenas 3,8, 4,4 e 5,7 pontos percentuais (no melhor caso), respectivamente. Assim, a arquitetura proposta se mostrou robusta, apresentando pequenos índices de degradação pois utilizou somente 2.707 registros, 1/8 do total, rotulados pelos curadores do GO, alcançando resultados próximos aos obtidos com aprendizado supervisionado.

A fim de representar graficamente essa diferença foi selecionado aleatoriamente, das 35 execuções realizadas, um resultado para C_4, C_6 e C_7. As Figuras 6.8, 6.9 e 6.10 ilustram os resultados obtidos para C_4, C_6 e C_7, respectivamente. Nesses gráficos, o eixo x corresponde aos conjuntos avaliados $(S_2, ..., S_8)$ e o eixo y corresponde à taxa de acerto das configurações analisadas.

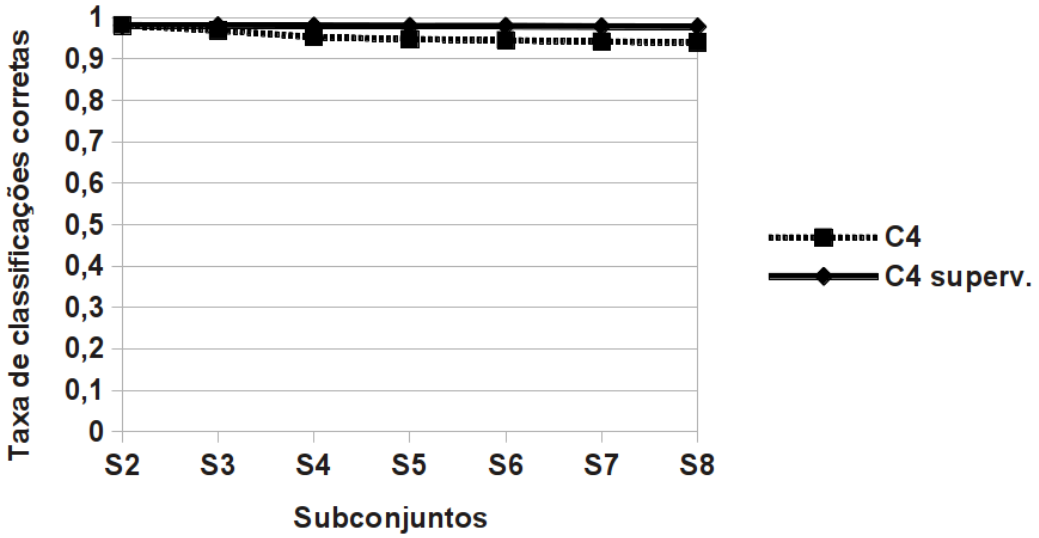

Figura 6.8: Comparação entre C_4 e C_4 supervisionado

Uma outra análise interessante é quando se compara a acurácia obtida com a utilização da configuração C_4 com as acurácias obtidas com a utilização das configurações C_6 e C_7 supervisionados. Nessa análise, as diferenças também foram pequenas, alcançando 0,6% quando comparado ao C_6 e 1,69% para o C_7. A mesma análise gráfica foi feita nessa comparação, como pode-se perceber na Figura 6.11, C_4 consegue resultados praticamente iguais aos obtidos por C_6 e C_7 supervisionados, mais precisamente, apenas 0,16 e 1,09 ponto percentual melhores, respectivamente, ao final das 8 iterações (selecionando um resultado dentre os 35 existentes).

A Figura 6.12 traz a comparação entre a configuração C_4 e J48/C_3 supervisionados, todas utilizando somente o classificador J48. Como pode-se perceber, os resultados encontrados para o C_4, que utiliza aprendizado semi-supervisionado, foi sempre superior ao J48 e C_3 supervisionados, chegando ao final classificando corretamente 7,75 e 6,85 pontos percentuais registros

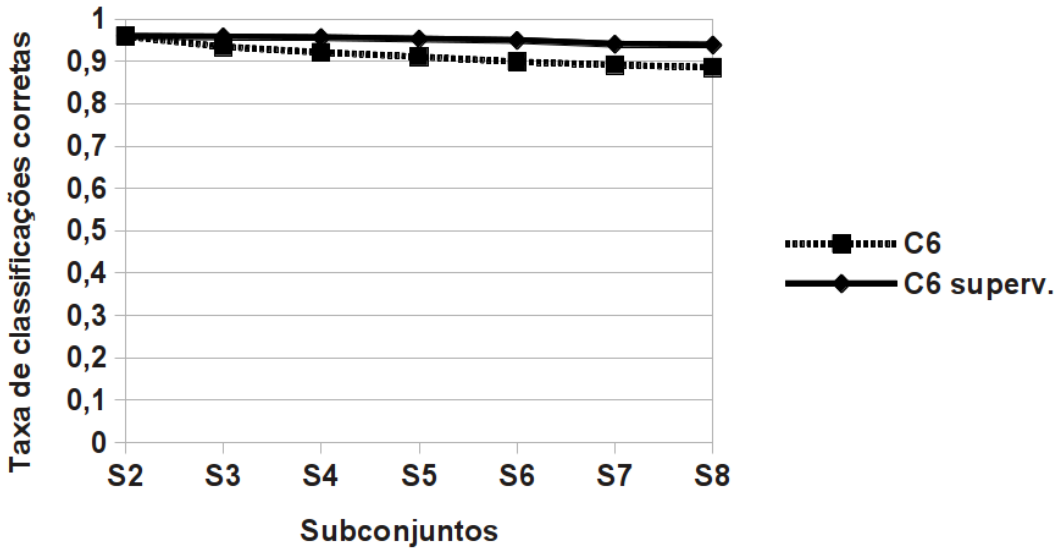

Figura 6.9: Comparação entre C_6 e C_6 supervisionado

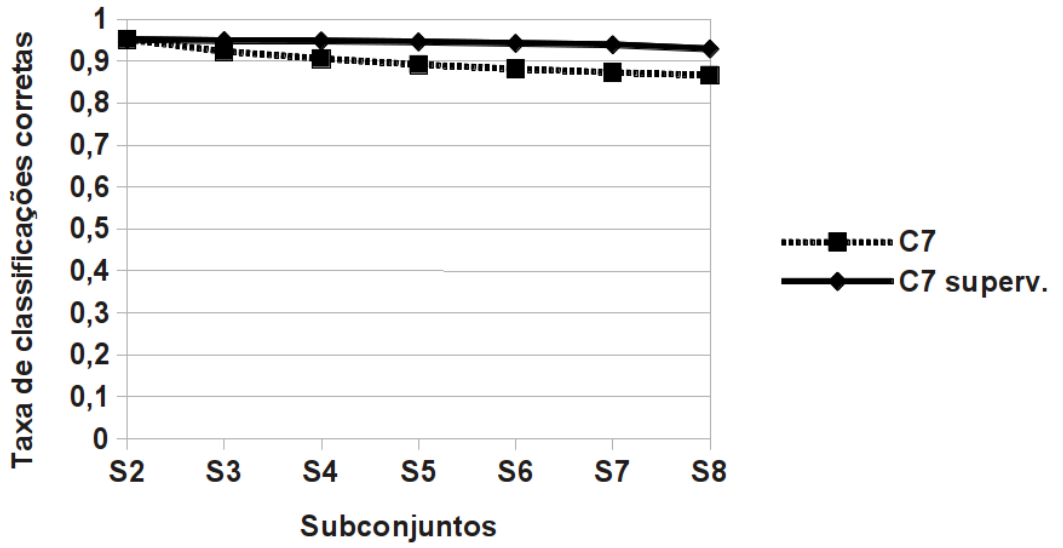

Figura 6.10: Comparação entre C_7 e C_7 supervisionado

a mais do que J48 e C_3, respectivamente. Dessa forma, o uso do J48 aplicado aos conjuntos de dados GOClasse e GOSubClasse dentro da arquitetura proposta levou à obtenção de resultados superiores aos encontrados para o J48 (utilizado de forma independente) e também para a configuração que aplicou o J48 ao conjunto de dados GOClasse dentro da arquitetura, ambos supervisionados.

O próximo capítulo, Capítulo 7, traz as conclusões que foram obtidas nessa pesquisa e os trabalhos futuros que serão desenvolvidos.

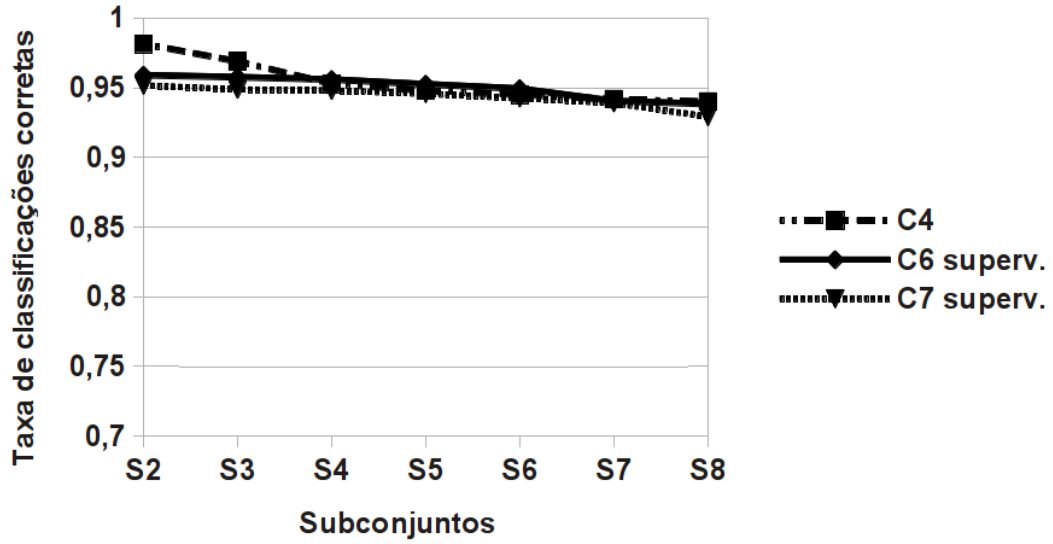

Figura 6.11: Comparação entre C_4 e $C_{6,7}$ supervisionados

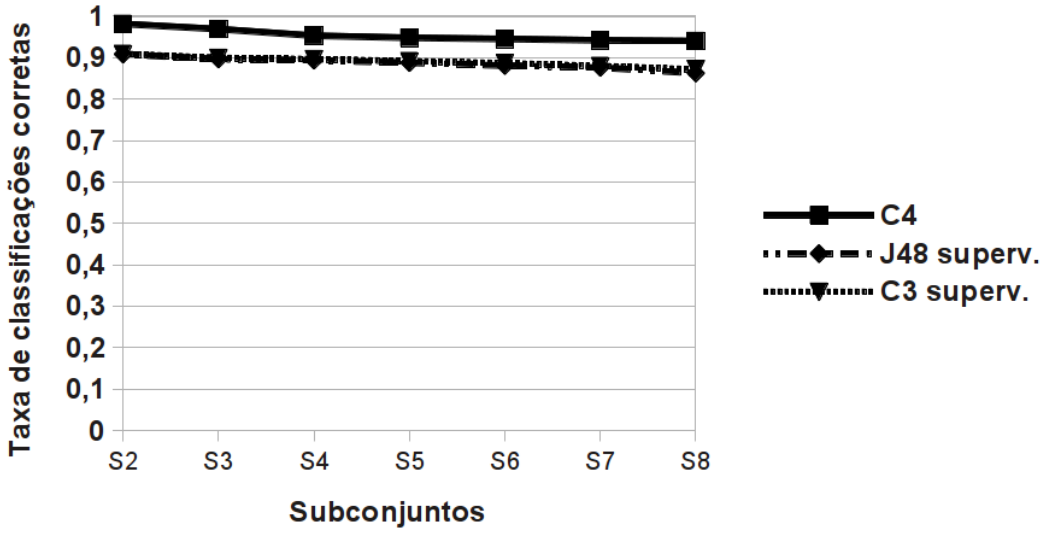

Figura 6.12: Comparação entre C_4, J48 e C_3 supervisionados

Capítulo 7

CONCLUSÕES E TRABALHOS FUTUROS

Atualmente, a nova realidade da pesquisa biológica é altamente dependente de biocuradores experientes, responsáveis por dar sentido ao dilúvio de dados gerados. Além disso, nos próximos anos os biocuradores enfrentarão grandes desafios. Estima-se que a geração de dados biológicos em 2020 será um milhão de vezes maior que a encontrada hoje. Dessa forma, a criação de métodos e ferramentas computacionais que auxiliem esses biocuradores é algo importantíssimo, permitindo aos biocuradores enfrentar os desafios atuais e os futuros relacionados ao gerenciamento de grandes volumes de dados, análise, interpretação e validação desses dados (BURGE et al., 2012).

Dessa forma, foi proposta nessa pesquisa uma arquitetura cujo principal objetivo é auxiliar os biocuradores do GO na classificação de novos termos. As configurações utilizadas apresentaram altos valores de classificação, variando entre 77,38 $\pm$ 0,25% (para a configuração C_2) à aproximadamente 94% (para a configuração C_4) de classificações corretas. Outro ponto importante está relacionado ao aumento da acucária ao se utilizar a arquitetura proposta. Comparando, por exemplo, o J48 e o C_4, o ganho foi superior a 10%, ultrapassando 2.000 termos classificados corretamente a mais.

Infelizmente, quando analisa-se o tempo gasto na classificação dos termos, o uso da arquitetura proposta aumentou significativamente, tendo alguns casos um aumento superior a 190%. Assim, o uso da arquitetura proposta, quando comparada com os classificadores individuais, aumentou drasticamente o tempo de processamento gasto em relação aos classificadores individuais. É importante ressaltar, entretanto, que tal aumento no tempo não tornou o método inviável (pois, os tempos de execução não chegaram a extrapolar três dezenas de minutos), mas, trouxe ganho considerável na precisão e na interpretabilidade dos resultados.

Com relação ao conhecimento gerado pelos *classificadores de entrada*, o aumento no tama-

nho da árvore gerada pelo uso da arquitetura proposta foi pequeno, sendo no melhor caso (J48 contra C_6), somente 4,52% maior (975 contra 1.019). Avaliando o tamanho das regras geradas pelo CEE (isso é, o número de antecedentes de cada regra), aplicado individualmente ou dentro da arquitetura, percebe-se que mínimas variações ocorreram, com diferenças de 0,04 (CEE contra C_6) e de 0,11 (CEE contra C_7). Dessa forma, na maioria das situações analisadas, o uso da arquitetura não gerou aumentos significativos no conhecimento gerado, isso é, na árvore de decisão e nas regras geradas pelos *classificadores de entrada*.

Quando analisa-se um classificador, a acurácia não é a única métrica analisada. Níveis de interpretabilidade e compreensibilidade também são importantes na avaliação de um classificador, isso é, o conhecimento gerado pelo mesmo também deve ser avaliado juntamente com a acurácia. Nessa pesquisa em particular, esse conhecimento gerado é extremamente importante, pois fornece ao biocurador do GO informações não somente sobre qual classe um novo termo pertence, mas também quais atributos são realmente importantes para determinada classe. Além disso, outras importantes informações em relação aos atributos são fornecidas, tais como intervalos numéricos. Essas informações supracitadas, auxiliam o biocurador a entender como os termos do GO estão rearranjados dentro do DAG, diminuindo assim, os erros na construção da ontologia final.

Dessa forma, a configuração C_6 conseguiu aliar acurácia, interpretabilidade e compreensibilidade do conhecimento gerado, obtendo bons valores de acurácia e gerando conhecimento na forma de árvores de decisão e regras SE-ENTÃO. O C_6, quando comparado ao C_4 (configuração com melhor valor de acurácia), obteve valores de acurácia apenas 5,68% menores e gerou uma árvore de decisão 41,67% menor. Além disso, as três regras geradas (selecionadas arbitrariamente, uma para cada classe) relaciona apenas 4 atributos (dos 34 existentes) do conjunto de dados analisado (GOClasse), restringindo a análise do biocurador do GO a um conjunto bem menor de atributos. Dessa forma, fica à escolha do biocurador do GO, selecionar e utilizar a configuração mais adequada, dentre a que possui melhor valor de acurácia ou aquela que agrega maiores níveis de interpretabilidade e compreensibilidade ao conhecimento gerado.

Um dos problemas inerentes quando utiliza-se o aprendizado semi-supervisionado está relacionado à degradação do desempenho do classificador, isso é, à cada iteração, o número de dados rotulados incorretamente aumenta, levando a uma queda na acurácia. Para tentar diminuir essa degradação, o NELL propôs o uso de acoplamentos entre os estratores de conhecimento. Nessa pesquisa, o acoplamento foi feito entre classificadores e o resultado foi positivo, obtendo resultados bem próximos (semi-supervisionado vs. supervisionado), com diferenças de 3,8%, 4,4% e 5,7%, para C_4, C_6 e C_7, respectivamente. Ao comparar o resultado obtido pelo C_4 semi-

supervisionado, com os obtidos para C_6 e C_7 supervisionados, o ambiente semi-supervionado consegue valores finais ligeiramente melhores que os demais, reforçando o que foi dito anteriormente sobre o uso dos acoplamentos. Um outro bom exemplo de como o uso de acoplamentos pode ser eficiente foi mostrado na Figura 6.12, onde C_4 semi-supervisionado foi comparado ao J48 e C_3, ambos supervisionados. Nessa situação pode-se perceber que com o uso de mais um *classificador de entrada* aplicado ao conjunto de dados GOSubClasse, dá ao ambiente semi-supervisionado condições de obter melhores valores de acurácia do que o J48 independente e C_3, ambos supervisionados.

Da forma como a arquitetura foi proposta, qualquer classificador pode ser utilizado como *classificador de entrada e saída*. Optou-se pelo CEE e J48 como *classificadores de entrada* pelo conhecimento gerado e pelo bom valor de acurácia apresentado. O NB foi utilizado como *classificador de saída* devido à sua simplicidade e, também, por apresentar bons valores de acucária.

Quando todos os dados produzidos ou publicados estiverem curados utilizando altos padrões de qualidade, além de disponíveis e acessíveis, a pesquisa biológica será conduzida de maneira bem diferente dos dias atuais. Os pesquisadores serão capazes de processar enormes quantidades de dados complexos mais rapidamente. Dessa forma, eles conseguirão gerar conhecimento sobre suas áreas de interesse mais rapidamente. Processando informações e gerando hipóteses através de métodos e ferramentas computacionais, os pesquisadores da área biológica poderão retornar às suas bancadas bem mais rapidamente para novos experimentos. Com essa nova forma de pesquisa, esses experimentos trarão novos conhecimentos, aumentando a especificidade, levando a um aumento exponencial do conhecimento gerado (HOWE et al., 2008).

O principal trabalho futuro a ser realizado é a busca para tornar a arquitetura proposta um método que utiliza todos os princípios do aprendizado de máquina sem-fim. O primeiro passo na busca por esse objetivo está relacionado à criação de novos acoplamentos, em tempo de execução e não somente utilizar os apresentados anteriormente. Neste sentido, explorar as relações entre termos da base GO é uma linha de atuação bastante promissora e muito pertinente quando se considera os acoplamentos do tipo *multi-view-agreement* (descritos no Capítulo 2). Um outro ponto está relacionado à ininterruptabilidade do aprendizado, onde a arquitetura proposta receberia, diariamente, dados oriundos de vários centros de pesquisa espalhados por todo o mundo e os classificaria em uma determinada classe, auxiliando os biocuradores do GO na construção da ontologia.

Devido aos resultados positivos alcançados pelo uso dos acoplamentos, um novo tipo de acoplamento poderia ser utilizando. Esse novo acoplamento utilizaria amostras negativas, isso

é, os registros pertencentes à classe BP, seriam utilizados no aprendizado sendo ¬ CC e/ou ¬ MF. O uso desse acoplamento envolvendo amostras negativas aumentará o número de *classificadores de entrada* e pode ser aplicado aos conjuntos de dados GOClasse e GOSubClasse. Para os *classificadores de entrada* utilizados nessa pesquisa (CEE e J48), o uso de amostras negativas seria aplicável somente para o J48, já que o CEE já trabalha internamente com amostras negativas.

Um outro trabalho a ser realizado está relacionado à aplicação de métodos de *graph mining* ao GO. *Graph mining* é uma nova área de pesquisa onde a busca do conhecimento é feita a partir de grafos (APPEL; Hruschka Jr., 2011). Dessa forma, métodos e ferramentas de *graph mining* serão aplicadas ao DAG do GO para a detecção de comunidades (LESKOVEC et al., 2009; FORTUNATO, 2010) e predição de arestas (HASAN et al., 2006; KASHIMA et al., 2009; KUNEGIS; LOMMATZSCH, 2009; Lü; ZHOU, 2009; ACAR; DUNLAVY; KOLDA, 2009). A detecção de comunidades pode trazer importantes informações sobre os grupos existentes dentro da ontologia do GO, agregando conhecimento para o biocurador, sendo uma importante ferramenta na classificação de novos termos. A predição de arestas pode ser uma importante ferramenta para os biocuradores do GO pois auxilia no processo de biocuragem, fornecendo predições sobre novas ligações dentro do DAG, relacionadas às termos já inseridos na ontologia e também em relação à novos termos. Na busca dessas comunidades e na predição de arestas, será utilizado o Prophet (APPEL; Hruschka Jr., 2011), proposto por Appel e Hruschka Jr.

REFERÊNCIAS

ACAR, E.; DUNLAVY, D. M.; KOLDA, T. G. Link prediction on evolving data using matrix and tensor factorizations. In: *ICDM Workshops*. [S.l.]: IEEE Computer Society, 2009. p. 262–269.

ALMEIDA, M. R. d.; BORéM, A.; FRANCO, G. R. *Biotecnologia e saúde*. [S.l.]: Universidade Federal de Viçosa, 2004.

AMARAL, L. R. d.; Hruschka Jr., E. R. Gene ontology classification: Building high-level knowledge using genetic algorithms. In: *IEEE Congress on Evolutionary Computation (CEC)*. [S.l.]: IEEE, 2010. p. 2610–2616.

AMARAL, L. R. d.; Hruschka Jr., E. R. Non-linear computational evolutionary environment (nlcee): Building high-level knowledge in complex biological databases. In: *Anais do Workshop: Data Mining in Functional Genomics and Proteomics: Current Trends and Future Directions*. [S.l.: s.n.], 2011. p. 73–82.

AMARAL, L. R. d.; Hruschka Jr., E. R. Never-ending learning principles in gene ontology classification using genetic algorithms. In: *IEEE Congress on Evolutionary Computation (CEC)*. [S.l.]: IEEE, 2012. p. 2292–2299.

AMARAL, L. R. d.; Hruschka Jr., E. R. Transgenic: an evolutionary algorithm operator. *Neurocomputing - in print*, 2013.

AMARAL, L. R. d. et al. Oncogenes classification measured by microarray using genetic algorithms. In: *Proceedings of the 26th IASTED International Conference on Artificial Intelligence and Applications*. Anaheim, CA, USA: ACTA Press, 2008. (AIA '08), p. 79–84.

APPEL, A. P.; Hruschka Jr., E. R. Centaurs - a component based framework to mine large graphs. *JIDM*, v. 2, n. 1, p. 19–26, 2011.

ASHBURNER, M. et al. Gene ontology: tool for the unification of biology. the gene ontology consortium. *Nature genetics*, Nature Publishing Group, v. 25, n. 1, p. 25–29, may 2000.

BASU, S. et al. Migenes: a searchable interspecies database of mitochondrial proteins curated using gene ontology annotation. *Bioinformatics*, v. 22, n. 4, p. 485–492, 2006.

BATEMAN, A. Curators of the world unite: the international society of biocuration. *Bioinformatics*, v. 26, n. 8, p. 991, 2010.

Baumgartner Jr., W. A. et al. Manual curation is not sufficient for annotation of genomic databases. In: *ISMB/ECCB (Supplement of Bioinformatics)*. [S.l.: s.n.], 2007. p. 41–48.

BLUM, A.; MITCHELL, T. Combining labeled and unlabeled data with co-training. In: *Proceedings of the Eleventh Annual Conference on Computational Learning Theory*. [S.l.]: ACM, 1998. p. 92–100.

BORéM, A.; SANTOS, F. R. d. *Biotecnologia Simplificada*. [S.l.]: Universidade Federal de Viçosa, 2004.

BOURNE, P. E.; MCENTYRE, J. Biocurators: Contributions to the world of science. *PLoS Computational Biology*, v. 2, n. 10, p. 1185, October 2006.

BURGE, S. et al. Biocurators and biocuration: surveying the 21st century challenges. *Database - Oxford*, 2012.

CARLSON, A. et al. Toward an architecture for never-ending language learning. In: *Proceedings of the Conference on Artificial Intelligence (AAAI)*. [S.l.]: AAAI Press, 2010. p. 1306–1313.

CARLSON, A. et al. Coupled semi-supervised learning for information extraction. In: *Proceedings of the Third ACM International Conference on Web Search and Data mining*. [S.l.]: ACM, 2010. p. 101–110.

CHAPELLE, O.; SCHöLKOPF, B.; ZIEN, A. *Semi-Supervised Learning*. [S.l.]: The MIT Press, 2006.

COSTA, N. M. B.; BORéM, A.; BARBOSA, C. O. *Alimentos transgênicos: saúde e segurança*. [S.l.]: Universidade Federal de Viçosa, 2005.

DEMPSTER, A. P.; LAIRD, N. M.; RUBIN, D. B. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society*, v. 39, n. 1, p. 1–38, 1977.

DIETTERICH, T. G. Machine-learning research: Four current directions. *AI Magazine*, v. 18, n. 4, p. 97–136, 1997.

DOMINGOS, P.; PAZZANI, M. On the optimality of the simple bayesian classifier under zero-one loss. *Machine Learning*, v. 29, p. 103–130, 1997.

DU, P. et al. From disease ontology to disease-ontology lite: statistical methods to adapt a general-purpose ontology for the test of gene-ontology associations. *Bioinformatics*, v. 25, n. 12, 2009.

DUDA, R. O.; HART, P. E. *Pattern Classification and Scene Analysis*. [S.l.]: John Willey & Sons, 1973.

FIDELIS, M. V.; LOPES, H. S.; FREITAS, A. A. Discovery comprehensible classification rules with a genetic algorithm. In: *IEEE Congress on Evolutionary Computation*. [S.l.]: IEEE, 2000. p. 805–810.

FORTUNATO, S. Community detection in graphs. *Physics Reports*, v. 486, n. 3-5, p. 75–174, 2010.

FRANK, A.; ASUNCION, A. *UCI - Machine Learning Repository*. 2010. Disponível em: <http://archive.ics.uci.edu/ml>.

FRANK, E. et al. Weka - a machine learning workbench for data mining. In: *Data Mining and Knowledge Discovery Handbook*. [S.l.]: Springer US, 2010. cap. 66, p. 1269–1277.

GOCONSORTIUM. The gene ontology: enhancements for 2011. *Nucleic Acids Research*, v. 40, n. Database-Issue, p. 559–564, 2012.

GOLDBERG, D. E. *Genetic Algorithms in Search, Optimization, and Machine Learning*. New York: Addison-Wesley, 1989.

GUYON, I.; ELISSEEFF, A. An introduction to variable and feature selection. *Journal of Machine Learning Research*, v. 3, p. 1157–1182, 2003.

HASAN, M. A. et al. Link prediction using supervised learning. In: *In Proceedings of SDM Workshop on Link Analysis, Counterterrorism and Security*. [S.l.: s.n.], 2006. p. 1–10.

HELLERSTEIN, J. L.; JAYRAM, T. S.; RISH, I. Recognizing end-user transactions in performance management. In: *AAAI - IAAI - Twelfth Conference on on Innovative Applications of Artificial Intelligence*. [S.l.]: AAAI Press / The MIT Press, 2000. p. 596–602.

HOWE, D. et al. Big data: The future of biocuration. *Nature*, v. 455, n. 4, p. 47–50, September 2008.

Hruschka Jr., E. R.; DUARTE, M. C.; NICOLETTI, M. C. Coupling as strategy for reducing concept-drift in never-ending learning environments. *Fundamenta Informaticae*, v. 124, n. 1-2, p. 47–61, 2013.

HUANG, Y. S.; SUEN, C. Y. A method of combining multiple experts for the recognition of unconstrained handwritten numerals. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, IEEE Computer Society, Los Alamitos, CA, USA, v. 17, p. 90–94, 1995. ISSN 0162-8828.

HUNTLEY, R. P. et al. Quickgo: a user tutorial for the web-based gene ontology browser. *Database - Oxford*, 2009.

JOSLYN, C. et al. The gene ontology categorizer. In: *ISMB/ECCB (Supplement of Bioinformatics)*. [S.l.: s.n.], 2004. p. 169–177.

KASHIMA, H. et al. Link propagation: A fast semi-supervised learning algorithm for link prediction. In: *SDM*. [S.l.]: SIAM, 2009. p. 1099–1110.

KITTLER, J. Combining classifiers: A theoretical framework. *Pattern Analysis and Applications*, v. 1, p. 18–27, 1998.

KOHAVI, R.; JOHN, G. H. "Wrappers for feature subset selection". *Artificial Intelligence*, v. 97, n. 1-2, p. 273–324, 1997.

KROGH, A.; VEDELSBY, J. Neural network ensembles, cross validation, and active learning. In: *Advances in Neural Information Processing Systems*. [S.l.]: MIT Press, 1995. p. 231–238.

KUNEGIS, J.; LOMMATZSCH, A. Learning spectral graph transformations for link prediction. In: *ICML*. [S.l.]: ACM, 2009. (ACM International Conference Proceeding Series), p. 561–568.

LATORRE, A. et al. Breast cancer biomarker selection using multiple offspring sampling. In: WARSAW, POLAND. *Proceedings of the ECML/PKDD. Workshop on Data Mining in Functional Genomics and Proteomics: Current Trends and Future Directions*. [S.l.], 2007.

LESKOVEC, J. et al. Community structure in large networks: Natural cluster sizes and the absence of large well-defined clusters. *Internet Mathematics*, v. 6, n. 1, p. 29–123, 2009.

LEVENICK, J. R. Inserting introns improves genetic algorithm success rate: Taking a cue from biology. In: *Proceedings of the Fourth International Conference on Genetic Algorithms*. [S.l.]: Morgan Kaufmann, 1991. p. 123–127.

LOPES, H. S.; COUTINHO, M. S.; LIMA, W. C. An evolutionary approach to simulate cognitive feedback learning in medical domain. In: *Genetic Algorithms and Fuzzy Logic Systems*. [S.l.]: World Scientific Publishing, 1997. v. 7, p. 193–207.

Lü, L.; ZHOU, T. Role of weak ties in link prediction of complex networks. In: *Proceedings of the 1st ACM international workshop on Complex networks meet information & knowledge management*. [S.l.]: ACM, 2009. (CNIKM '09), p. 55–58.

MARKOWITZ, V. M. et al. Img er: a system for microbial genome annotation expert review and curation. *Bioinformatics*, v. 25, n. 17, p. 2271–2278, 2009.

MCCALLUM, A.; NIGAM, K. A comparison of event models for naive bayes text classification. In: *AAAI Workshop on Learning for Text Categorization*. [S.l.: s.n.], 1998. p. 41–48.

MITCHELL, M. *An introduction to genetic algorithms*. [S.l.]: MIT Press, 1999.

MITCHELL, T. M. *Machine Learning*. [S.l.]: McGraw-Hill, 1997.

PARK, J. et al. Cvmango, a method for leveraging computational predictions to improve literature-based gene ontology annotations. *Database - Oxford*, 2012.

PEDERSEN, T. A decision tree of bigrams is an accurate predictor of word sense. In: *In Proceedings of the Second Annual Meeting of the North American Chapter of the Association for Computational Linguistics*. [S.l.: s.n.], 2001. p. 79–86.

QUINLAN, J. R. *C4.5: Programs for Machine Learning*. [S.l.]: Morgan Kaufmann Publishers, 1993.

RICHARDS, A. J. et al. Assessing the functional coherence of gene sets with metrics based on the gene ontology graph. *Bioinformatics [ISMB]*, v. 26, n. 12, p. 79–87, 2010.

ROSS, D. T. et al. Systematic variation in gene expression patterns in human cancer cell lines. *Nature Genetics*, v. 24, n. 3, p. 227–35, 2000,.

SALIMI, N.; VITA, R. The biocurator: Connecting and enhancing scientific data. *PLoS Computational Biology*, v. 2, n. 10, p. 1190, October 2006.

SERINGHAUS, M.; GERSTEIN, M. Publishing perishing? towards tomorrow's information architecture. *BMC - Bioinformatics*, v. 8, n. 1, p. 17–21, 2007. ISSN 1471-2105.

SILVA, R. G. O.; RIBEIRO, M. W. S. de; AMARAL, L. R. d. Building high level knowledge from high dimensionality biological dataset (nci60) using genetic algorithms and feature selection strategies. In: *IEEE Congress on Evolutionary Computation*. [S.l.]: IEEE, 2013. p. 578–583.

SMID, M.; DORSSERS, L. C. J. Go-mapper: functional analysis of gene expression data using the expression level as a score to evaluate gene ontology terms. *Bioinformatics*, v. 20, n. 16, p. 2618–2625, 2004.

TAN, P.-N.; STEINBACH, M.; KUMAR, V. *Introduction to Data Mining*. [S.l.]: Editora Ciência Moderna, 2009.

TAO, Y. et al. Information theory applied to the sparse gene ontology annotation network to predict novel gene function. In: *ISMB/ECCB (Supplement of Bioinformatics)*. [S.l.: s.n.], 2007. p. 529–538.

VERMA, S.; Hruschka Jr., E. R. Coupled bayesian sets algorithm for semi-supervised learning and information extraction. In: *Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECMLPKDD 2012)*. [S.l.]: Association for Computing Machinery, 2012. p. 307–322.

WOLPERT, D. H. Stacked generalization. *Neural Networks*, v. 5, n. 2, p. 241–260, 1992.

YEH, I. et al. Knowledge acquisition, consistency checking and concurrency control for gene ontology (go). *Bioinformatics*, v. 19, n. 2, p. 241–248, 2003.

ZHU, X. *Semi-Supervised Learning Literature Survey*. [S.l.], 2005.

GLOSSÁRIO

AG - Algoritmo genético

AM - Aprendizado de Máquina

BP - Processo biológico

CC - Componente celular

CEE - Computational Evolutionary Environment

CvManGO - Computational versus Manual GO annotation

DAG - Grafo direcionado acíclico

DNA - Ácido desoxirribonucléico

EM - Expectation-maximization

FA - Função de avaliação

GO - Gene Ontology

IA - Inteligência Artificial

IEA - Anotações inferidas eletrônicamente

MD - Mineração de dados

MF - Função molecular

NB - Naive Bayes

NELL - Never-Ending Language Learner

NLCEE - Non-Linear Computational Evolutionary Environment

OBO - Open Biological and Biomedical Ontologies

SCI - Sistemas Computacionais Inteligentes

SE - Sensibilidade

SP - Especificidade

SQL - Linguagem de consulta estruturada

Apendice A

CONSULTAS SQL UTILIZADAS NA PESQUISA

A.1 Consulta SQL utilizada na obtenção dos registros relacionados às classes BP, CC e MF

insert into tbschema(intTRMid, strTRMname, strTRMterm_type,
intQtdeTermosAdjEhPai, intQtdeTermosAdjEhFilho, intDistFilhoMax, intQtdeFilhos,
fltDistMediaFilhos, fltDistDesvPadFilhos, intDistPaiMax, intQtdePais, fltDistMediaPais,
fltDistDesvPadPais, intNroSpecies, intNroGeneProduct, fltMediaProductCountSpecies,
intQtdeSeqs, fltPorcAlanine, fltPorcArginine, fltPorcAsparagine, fltPorcAsparticAcid,
fltPorcCysteine, fltPorcGlutamicAcid, fltPorcGlutamine, fltPorcGlycine, fltPorcHistidine,
fltPorcIsoleucine, fltPorcLeucine, fltPorcLysine, fltPorcMethionine, fltPorcPhenylalanine,
fltPorcProline, fltPorcSerine, fltPorcThreonine, fltPorcTryptophan, fltPorcTyrosine,
fltPorcValine)

select TRM.id as TRMid,
TRM.name as TRMname,
TRM.term_type as TRMterm_type,

(select count(T2T2.term2_id)
from term2term T2T2
where T2T2.term1_id = TRM.id) as intQtdeTermosAdjEhPai,

(select count(T2T2.term1_id)
from term2term T2T2
where T2T2.term2_id = TRM.id) as intQtdeTermosAdjEhFilho,

(select max(GPH.distance)

```
from graph_path GPH
where GPH.term1_id = TRM.id) as intDistFilhoMax,

    (select count(GPH.distance)
from graph_path GPH
where GPH.term1_id = TRM.id) as intQtdeFilhos,

    (select avg(GPH.distance)
from graph_path GPH
where GPH.term1_id = TRM.id) as fltDistMediaFilhos,

    (select std(GPH.distance)
from graph_path GPH
where GPH.term1_id = TRM.id) as fltDistDesvPadFilhos,

    – Pais (select max(GPH.distance)
from graph_path GPH
where GPH.term2_id = TRM.id) as intDistPaiMax,

    (select count(GPH.distance)
from graph_path GPH
where GPH.term2_id = TRM.id) as intQtdePais,

    (select avg(GPH.distance)
from graph_path GPH
where GPH.term2_id = TRM.id) as fltDistMediaPais,

    (select std(GPH.distance)
from graph_path GPH
where GPH.term2_id = TRM.id) as fltDistDesvPadPais,

    (select count(species_id)
from gene_product_count GPC
where GPC.term_id = TRM.id) as intNroSpecies,

    (select sum(GPC.product_count)
from gene_product_count GPC
where GPC.term_id = TRM.id) as intNroGeneProduct,

    (select sum(GPC.product_count)/count(species_id)
from gene_product_count GPC
where GPC.term_id = TRM.id) as fltMediaProductCountSpecies,
```

```
    (select count(SEQ.seq)
from gene_product_seq GPS, association ASS, seq SEQ
where GPS.gene_product_id = ASS.gene_product_id
and GPS.seq_id = SEQ.id
and ASS.term_id = TRM.id) as intQtdeSeqs,

    (select (sum((SELECT
LENGTH(SEQ.seq)-LENGTH(REPLACE(SEQ.seq,'A',''))))/sum(SEQ.seq_len))
from gene_product_seq GPS, association ASS, seq SEQ
where GPS.gene_product_id = ASS.gene_product_id
and GPS.seq_id = SEQ.id
and ASS.term_id = TRM.id) as fltPorcAlanine,

    (select (sum((SELECT
LENGTH(SEQ.seq)-LENGTH(REPLACE(SEQ.seq,'R',''))))/sum(SEQ.seq_len))
from gene_product_seq GPS, association ASS, seq SEQ
where GPS.gene_product_id = ASS.gene_product_id
and GPS.seq_id = SEQ.id
and ASS.term_id = TRM.id) as fltPorcArginine,

    (select (sum((SELECT
LENGTH(SEQ.seq)-LENGTH(REPLACE(SEQ.seq,'N',''))))/sum(SEQ.seq_len))
from gene_product_seq GPS, association ASS, seq SEQ
where GPS.gene_product_id = ASS.gene_product_id
and GPS.seq_id = SEQ.id
and ASS.term_id = TRM.id) as fltPorcAsparagine,

    (select (sum((SELECT
LENGTH(SEQ.seq)-LENGTH(REPLACE(SEQ.seq,'D',''))))/sum(SEQ.seq_len))
from gene_product_seq GPS, association ASS, seq SEQ
where GPS.gene_product_id = ASS.gene_product_id
and GPS.seq_id = SEQ.id
and ASS.term_id = TRM.id) as fltPorcAsparticAcid,

    (select (sum((SELECT
LENGTH(SEQ.seq)-LENGTH(REPLACE(SEQ.seq,'C',''))))/sum(SEQ.seq_len))
from gene_product_seq GPS, association ASS, seq SEQ
where GPS.gene_product_id = ASS.gene_product_id
and GPS.seq_id = SEQ.id
```

```
and ASS.term_id = TRM.id) as fltPorcCysteine,

    (select (sum((SELECT
LENGTH(SEQ.seq)-LENGTH(REPLACE(SEQ.seq,'E',''))))/sum(SEQ.seq_len))
from gene_product_seq GPS, association ASS, seq SEQ
where GPS.gene_product_id = ASS.gene_product_id
and GPS.seq_id = SEQ.id
and ASS.term_id = TRM.id) as fltPorcGlutamicAcid,

    (select (sum((SELECT
LENGTH(SEQ.seq)-LENGTH(REPLACE(SEQ.seq,'Q',''))))/sum(SEQ.seq_len))
from gene_product_seq GPS, association ASS, seq SEQ
where GPS.gene_product_id = ASS.gene_product_id
and GPS.seq_id = SEQ.id
and ASS.term_id = TRM.id) as fltPorcGlutamine,

    (select (sum((SELECT
LENGTH(SEQ.seq)-LENGTH(REPLACE(SEQ.seq,'G',''))))/sum(SEQ.seq_len))
from gene_product_seq GPS, association ASS, seq SEQ
where GPS.gene_product_id = ASS.gene_product_id
and GPS.seq_id = SEQ.id
and ASS.term_id = TRM.id) as fltPorcGlycine,

    (select (sum((SELECT
LENGTH(SEQ.seq)-LENGTH(REPLACE(SEQ.seq,'H',''))))/sum(SEQ.seq_len))
from gene_product_seq GPS, association ASS, seq SEQ
where GPS.gene_product_id = ASS.gene_product_id
and GPS.seq_id = SEQ.id
and ASS.term_id = TRM.id) as fltPorcHistidine,

    (select (sum((SELECT
LENGTH(SEQ.seq)-LENGTH(REPLACE(SEQ.seq,'I',''))))/sum(SEQ.seq_len))
from gene_product_seq GPS, association ASS, seq SEQ
where GPS.gene_product_id = ASS.gene_product_id
and GPS.seq_id = SEQ.id
and ASS.term_id = TRM.id) as fltPorcIsoleucine,

    (select (sum((SELECT
LENGTH(SEQ.seq)-LENGTH(REPLACE(SEQ.seq,'L',''))))/sum(SEQ.seq_len))
```

```
from gene_product_seq GPS, association ASS, seq SEQ
where GPS.gene_product_id = ASS.gene_product_id
and GPS.seq_id = SEQ.id
and ASS.term_id = TRM.id) as fltPorcLeucine,

    (select (sum((SELECT
LENGTH(SEQ.seq)-LENGTH(REPLACE(SEQ.seq,'K',''))))/sum(SEQ.seq_len))
from gene_product_seq GPS, association ASS, seq SEQ
where GPS.gene_product_id = ASS.gene_product_id
and GPS.seq_id = SEQ.id
and ASS.term_id = TRM.id) as fltPorcLysine,

    (select (sum((SELECT
LENGTH(SEQ.seq)-LENGTH(REPLACE(SEQ.seq,'M',''))))/sum(SEQ.seq_len))
from gene_product_seq GPS, association ASS, seq SEQ
where GPS.gene_product_id = ASS.gene_product_id
and GPS.seq_id = SEQ.id
and ASS.term_id = TRM.id) as fltPorcMethionine,

    (select (sum((SELECT
LENGTH(SEQ.seq)-LENGTH(REPLACE(SEQ.seq,'F',''))))/sum(SEQ.seq_len))
from gene_product_seq GPS, association ASS, seq SEQ
where GPS.gene_product_id = ASS.gene_product_id
and GPS.seq_id = SEQ.id
and ASS.term_id = TRM.id) as fltPorcPhenylalanine,

    (select (sum((SELECT
LENGTH(SEQ.seq)-LENGTH(REPLACE(SEQ.seq,'P',''))))/sum(SEQ.seq_len))
from gene_product_seq GPS, association ASS, seq SEQ
where GPS.gene_product_id = ASS.gene_product_id
and GPS.seq_id = SEQ.id
and ASS.term_id = TRM.id) as fltPorcProline,

    (select (sum((SELECT
LENGTH(SEQ.seq)-LENGTH(REPLACE(SEQ.seq,'S',''))))/sum(SEQ.seq_len))
from gene_product_seq GPS, association ASS, seq SEQ
where GPS.gene_product_id = ASS.gene_product_id
and GPS.seq_id = SEQ.id
and ASS.term_id = TRM.id) as fltPorcSerine,
```

```
(select (sum((SELECT
LENGTH(SEQ.seq)-LENGTH(REPLACE(SEQ.seq,'T',''))))/sum(SEQ.seq_len))
from gene_product_seq GPS, association ASS, seq SEQ
where GPS.gene_product_id = ASS.gene_product_id
and GPS.seq_id = SEQ.id
and ASS.term_id = TRM.id) as fltPorcThreonine,

(select (sum((SELECT
LENGTH(SEQ.seq)-LENGTH(REPLACE(SEQ.seq,'W',''))))/sum(SEQ.seq_len))
from gene_product_seq GPS, association ASS, seq SEQ
where GPS.gene_product_id = ASS.gene_product_id
and GPS.seq_id = SEQ.id
and ASS.term_id = TRM.id) as fltPorcTryptophan,

(select (sum((SELECT
LENGTH(SEQ.seq)-LENGTH(REPLACE(SEQ.seq,'Y',''))))/sum(SEQ.seq_len))
from gene_product_seq GPS, association ASS, seq SEQ
where GPS.gene_product_id = ASS.gene_product_id
and GPS.seq_id = SEQ.id
and ASS.term_id = TRM.id) as fltPorcTyrosine,

(select (sum((SELECT
LENGTH(SEQ.seq)-LENGTH(REPLACE(SEQ.seq,'V',''))))/sum(SEQ.seq_len))
from gene_product_seq GPS, association ASS, seq SEQ
where GPS.gene_product_id = ASS.gene_product_id
and GPS.seq_id = SEQ.id
and ASS.term_id = TRM.id) as fltPorcValine

from liteseqdb.term TRM
where ((TRM.term_type = 'molecular_function') or (TRM.term_type = 'cellular_component')
or (TRM.term_type = 'biological_process'))
having fltPorcValine is not null
order by TRM.term_type
```

A.2 Consultas SQL utilizadas na obtenção dos registros relacionados às subclasses de BP, CC e MF

A.2.1 Subclasse de BP

insert into tbschema (intTRMid, strTRMname, intQtdeTermosAdjEhPai, intQtdeTermosAdjEhFilho, intDistFilhoMax, intQtdeFilhos, fltDistMediaFilhos, fltDistDesvPadFilhos, intDistPaiMax, intQtdePais, fltDistMediaPais, intNroSpecies, intNroGeneProduct, fltMediaProductCountSpecies, intQtdeSeqs, fltPorcAlanine, fltPorcArginine, fltPorcAsparagine, fltPorcAsparticAcid, fltPorcCysteine, fltPorcGlutamicAcid, fltPorcGlutamine, fltPorcGlycine, fltPorcHistidine, fltPorcIsoleucine, fltPorcLeucine, fltPorcLysine, fltPorcMethionine, fltPorcPhenylalanine, fltPorcProline, fltPorcSerine, fltPorcThreonine, fltPorcTryptophan, fltPorcTyrosine,fltPorcValine, strTRMterm_type)

select distinct GPH.term2_id as intTRMid,

(select TRM3.name
from term TRM3
where TRM3.id = GPH.term2_id) as strTRMname,

(select count(T2T2.term2_id)
from term2term T2T2
where T2T2.term1_id = GPH.term2_id) as intQtdeTermosAdjEhPai,

(select count(T2T2.term1_id)
from term2term T2T2
where T2T2.term2_id = GPH.term2_id) as intQtdeTermosAdjEhFilho ,

(select max(GPH2.distance)
from graph_path GPH2
where GPH2.term1_id = GPH.term2_id) as intDistFilhoMax ,

(select count(GPH2.distance)
from graph_path GPH2
where GPH2.term1_id = GPH.term2_id) as intQtdeFilhos ,

(select avg(GPH2.distance)
from graph_path GPH2
where GPH2.term1_id = GPH.term2_id) as fltDistMediaFilhos ,

```
    (select std(GPH2.distance)
from graph_path GPH2
where GPH2.term1_id = GPH.term2_id) as fltDistDesvPadFilhos ,

    (select max(GPH2.distance)
from graph_path GPH2
where GPH2.term2_id = GPH.term2_id) as intDistPaiMax ,

    (select count(GPH2.distance)
from graph_path GPH2
where GPH2.term2_id = GPH.term2_id) as intQtdePais ,

    (select avg(GPH2.distance)
from graph_path GPH2
where GPH2.term2_id = GPH.term2_id) as fltDistMediaPais ,

    (select count(GPC.species_id)
from gene_product_count GPC
where GPC.term_id = GPH.term2_id) as intNroSpecies ,

    (select sum(GPC.product_count)
from gene_product_count GPC
where GPC.term_id = GPH.term2_id) as intNroGeneProduct,

    (select sum(GPC.product_count)/count(GPC.species_id)
from gene_product_count GPC
where GPC.term_id = GPH.term2_id) as fltMediaProductCountSpecies,

    (select count(SEQ.seq)
from gene_product_seq GPS, association ASS, seq SEQ
where GPS.gene_product_id = ASS.gene_product_id
and GPS.seq_id = SEQ.id
and ASS.term_id = GPH.term2_id) as intQtdeSeqs,

    (select (sum((SELECT
LENGTH(SEQ.seq)-LENGTH(REPLACE(SEQ.seq,'A',''))))/sum(SEQ.seq_len))
from gene_product_seq GPS, association ASS, seq SEQ
where GPS.gene_product_id = ASS.gene_product_id
and GPS.seq_id = SEQ.id
and ASS.term_id = GPH.term2_id) as fltPorcAlanine ,
```

```
(select (sum((SELECT
LENGTH(SEQ.seq)-LENGTH(REPLACE(SEQ.seq,'R',''))))/sum(SEQ.seq_len))
from gene_product_seq GPS, association ASS, seq SEQ
where GPS.gene_product_id = ASS.gene_product_id
and GPS.seq_id = SEQ.id
and ASS.term_id = GPH.term2_id) as fltPorcArginine,

(select (sum((SELECT
LENGTH(SEQ.seq)-LENGTH(REPLACE(SEQ.seq,'N',''))))/sum(SEQ.seq_len))
from gene_product_seq GPS, association ASS, seq SEQ
where GPS.gene_product_id = ASS.gene_product_id
and GPS.seq_id = SEQ.id
and ASS.term_id = GPH.term2_id) as fltPorcAsparagine,

(select (sum((SELECT
LENGTH(SEQ.seq)-LENGTH(REPLACE(SEQ.seq,'D',''))))/sum(SEQ.seq_len))
from gene_product_seq GPS, association ASS, seq SEQ
where GPS.gene_product_id = ASS.gene_product_id
and GPS.seq_id = SEQ.id
and ASS.term_id = GPH.term2_id) as fltPorcAsparticAcid,

(select (sum((SELECT
LENGTH(SEQ.seq)-LENGTH(REPLACE(SEQ.seq,'C',''))))/sum(SEQ.seq_len))
from gene_product_seq GPS, association ASS, seq SEQ
where GPS.gene_product_id = ASS.gene_product_id
and GPS.seq_id = SEQ.id
and ASS.term_id = GPH.term2_id) as fltPorcCysteine,

(select (sum((SELECT
LENGTH(SEQ.seq)-LENGTH(REPLACE(SEQ.seq,'E',''))))/sum(SEQ.seq_len))
from gene_product_seq GPS, association ASS, seq SEQ
where GPS.gene_product_id = ASS.gene_product_id
and GPS.seq_id = SEQ.id
and ASS.term_id = GPH.term2_id) as fltPorcGlutamicAcid,

(select (sum((SELECT
LENGTH(SEQ.seq)-LENGTH(REPLACE(SEQ.seq,'Q',''))))/sum(SEQ.seq_len))
from gene_product_seq GPS, association ASS, seq SEQ
where GPS.gene_product_id = ASS.gene_product_id
```

```
and GPS.seq_id = SEQ.id
and ASS.term_id = GPH.term2_id) as fltPorcGlutamine,

    (select (sum((SELECT
LENGTH(SEQ.seq)-LENGTH(REPLACE(SEQ.seq,'G',''))))/sum(SEQ.seq_len))
from gene_product_seq GPS, association ASS, seq SEQ
where GPS.gene_product_id = ASS.gene_product_id
and GPS.seq_id = SEQ.id
and ASS.term_id = GPH.term2_id) as fltPorcGlycine,

    (select (sum((SELECT
LENGTH(SEQ.seq)-LENGTH(REPLACE(SEQ.seq,'H',''))))/sum(SEQ.seq_len))
from gene_product_seq GPS, association ASS, seq SEQ
where GPS.gene_product_id = ASS.gene_product_id
and GPS.seq_id = SEQ.id
and ASS.term_id = GPH.term2_id) as fltPorcHistidine,

    (select (sum((SELECT
LENGTH(SEQ.seq)-LENGTH(REPLACE(SEQ.seq,'I',''))))/sum(SEQ.seq_len))
from gene_product_seq GPS, association ASS, seq SEQ
where GPS.gene_product_id = ASS.gene_product_id
and GPS.seq_id = SEQ.id
and ASS.term_id = GPH.term2_id) as fltPorcIsoleucine,

    (select (sum((SELECT
LENGTH(SEQ.seq)-LENGTH(REPLACE(SEQ.seq,'L',''))))/sum(SEQ.seq_len))
from gene_product_seq GPS, association ASS, seq SEQ
where GPS.gene_product_id = ASS.gene_product_id
and GPS.seq_id = SEQ.id
and ASS.term_id = GPH.term2_id) as fltPorcLeucine,

    (select (sum((SELECT
LENGTH(SEQ.seq)-LENGTH(REPLACE(SEQ.seq,'K',''))))/sum(SEQ.seq_len))
from gene_product_seq GPS, association ASS, seq SEQ
where GPS.gene_product_id = ASS.gene_product_id
and GPS.seq_id = SEQ.id
and ASS.term_id = GPH.term2_id) as fltPorcLysine,

    (select (sum((SELECT
```

```
LENGTH(SEQ.seq)-LENGTH(REPLACE(SEQ.seq,'M',''))))/sum(SEQ.seq_len))
from gene_product_seq GPS, association ASS, seq SEQ
where GPS.gene_product_id = ASS.gene_product_id
and GPS.seq_id = SEQ.id
and ASS.term_id = GPH.term2_id) as fltPorcMethionine,

    (select (sum((SELECT
LENGTH(SEQ.seq)-LENGTH(REPLACE(SEQ.seq,'F',''))))/sum(SEQ.seq_len))
from gene_product_seq GPS, association ASS, seq SEQ
where GPS.gene_product_id = ASS.gene_product_id
and GPS.seq_id = SEQ.id
and ASS.term_id = GPH.term2_id) as fltPorcPhenylalanine,

    (select (sum((SELECT
LENGTH(SEQ.seq)-LENGTH(REPLACE(SEQ.seq,'P',''))))/sum(SEQ.seq_len))
from gene_product_seq GPS, association ASS, seq SEQ
where GPS.gene_product_id = ASS.gene_product_id
and GPS.seq_id = SEQ.id
and ASS.term_id = GPH.term2_id) as fltPorcProline,

    (select (sum((SELECT
LENGTH(SEQ.seq)-LENGTH(REPLACE(SEQ.seq,'S',''))))/sum(SEQ.seq_len))
from gene_product_seq GPS, association ASS, seq SEQ
where GPS.gene_product_id = ASS.gene_product_id
and GPS.seq_id = SEQ.id
and ASS.term_id = GPH.term2_id) as fltPorcSerine,

    (select (sum((SELECT
LENGTH(SEQ.seq)-LENGTH(REPLACE(SEQ.seq,'T',''))))/sum(SEQ.seq_len))
from gene_product_seq GPS, association ASS, seq SEQ
where GPS.gene_product_id = ASS.gene_product_id
and GPS.seq_id = SEQ.id
and ASS.term_id = GPH.term2_id) as fltPorcThreonine,

    (select (sum((SELECT
LENGTH(SEQ.seq)-LENGTH(REPLACE(SEQ.seq,'W',''))))/sum(SEQ.seq_len))
from gene_product_seq GPS, association ASS, seq SEQ
where GPS.gene_product_id = ASS.gene_product_id
and GPS.seq_id = SEQ.id
```

```
and ASS.term_id = GPH.term2_id) as fltPorcTryptophan,

    (select (sum((SELECT
LENGTH(SEQ.seq)-LENGTH(REPLACE(SEQ.seq,'Y',''))))/sum(SEQ.seq_len))
from gene_product_seq GPS, association ASS, seq SEQ
where GPS.gene_product_id = ASS.gene_product_id
and GPS.seq_id = SEQ.id
and ASS.term_id = GPH.term2_id) as fltPorcTyrosine ,

    (select (sum((SELECT
LENGTH(SEQ.seq)-LENGTH(REPLACE(SEQ.seq,'V',''))))/sum(SEQ.seq_len))
from gene_product_seq GPS, association ASS, seq SEQ
where GPS.gene_product_id = ASS.gene_product_id
and GPS.seq_id = SEQ.id
and ASS.term_id = GPH.term2_id) as fltPorcValine,

    (CONCAT('BP-',(select TRM3.name
from term TRM3
where TRM3.id = GPH.term1_id))) as strTRMterm_type

    from term TRM, graph_path GPH
where TRM.id = GPH.term1_id
and TRM.id in (
select GPH2.term2_id
from term TRM2, graph_path GPH2
where TRM2.id = GPH2.term1_id
and TRM2.term_type = 'biological_process'
and TRM2.name = 'biological_process'
and GPH2.distance = 1)

    and GPH.term1_id < > GPH.term2_id
having fltPorcValine is not null
```

A.2.2 Subclasse de CC

```
    insert IGNORE into SCGO16032013.tbschema (intTRMid, strTRMname,
intQtdeTermosAdjEhPai, intQtdeTermosAdjEhFilho, intDistFilhoMax, intQtdeFilhos,
fltDistMediaFilhos, fltDistDesvPadFilhos, intDistPaiMax, intQtdePais, fltDistMediaPais,
```

intNroSpecies, intNroGeneProduct, fltMediaProductCountSpecies, intQtdeSeqs, fltPorcAlanine, fltPorcArginine, fltPorcAsparagine, fltPorcAsparticAcid, fltPorcCysteine, fltPorcGlutamicAcid, fltPorcGlutamine, fltPorcGlycine, fltPorcHistidine, fltPorcIsoleucine, fltPorcLeucine, fltPorcLysine, fltPorcMethionine, fltPorcPhenylalanine, fltPorcProline, fltPorcSerine, fltPorcThreonine, fltPorcTryptophan, fltPorcTyrosine, fltPorcValine, strTRMterm_type)

select distinct GPH.term2_id as intTRMid,

(select TRM3.name
from term TRM3
where TRM3.id = GPH.term2_id) as strTRMname,

(select count(T2T2.term2_id)
from term2term T2T2
where T2T2.term1_id = GPH.term2_id) as intQtdeTermosAdjEhPai,

(select count(T2T2.term1_id)
from term2term T2T2
where T2T2.term2_id = GPH.term2_id) as intQtdeTermosAdjEhFilho ,

(select max(GPH2.distance)
from graph_path GPH2
where GPH2.term1_id = GPH.term2_id) as intDistFilhoMax ,

(select count(GPH2.distance)
from graph_path GPH2
where GPH2.term1_id = GPH.term2_id) as intQtdeFilhos ,

(select avg(GPH2.distance)
from graph_path GPH2
where GPH2.term1_id = GPH.term2_id) as fltDistMediaFilhos ,

(select std(GPH2.distance)
from graph_path GPH2
where GPH2.term1_id = GPH.term2_id) as fltDistDesvPadFilhos ,

(select max(GPH2.distance)
from graph_path GPH2
where GPH2.term2_id = GPH.term2_id) as intDistPaiMax ,

(select count(GPH2.distance)

```
from graph_path GPH2
where GPH2.term2_id = GPH.term2_id) as intQtdePais ,

    (select avg(GPH2.distance)
from graph_path GPH2
where GPH2.term2_id = GPH.term2_id) as fltDistMediaPais ,

    (select count(GPC.species_id)
from gene_product_count GPC
where GPC.term_id = GPH.term2_id) as intNroSpecies ,

    (select sum(GPC.product_count)
from gene_product_count GPC
where GPC.term_id = GPH.term2_id) as intNroGeneProduct,

    (select sum(GPC.product_count)/count(GPC.species_id)
from gene_product_count GPC
where GPC.term_id = GPH.term2_id) as fltMediaProductCountSpecies,

    (select count(SEQ.seq)
from gene_product_seq GPS, association ASS, seq SEQ
where GPS.gene_product_id = ASS.gene_product_id
and GPS.seq_id = SEQ.id
and ASS.term_id = GPH.term2_id) as intQtdeSeqs,

    (select (sum((SELECT
LENGTH(SEQ.seq)-LENGTH(REPLACE(SEQ.seq,'A',''))))/sum(SEQ.seq_len))
from gene_product_seq GPS, association ASS, seq SEQ
where GPS.gene_product_id = ASS.gene_product_id
and GPS.seq_id = SEQ.id
and ASS.term_id = GPH.term2_id) as fltPorcAlanine,

    (select (sum((SELECT
LENGTH(SEQ.seq)-LENGTH(REPLACE(SEQ.seq,'R',''))))/sum(SEQ.seq_len))
from gene_product_seq GPS, association ASS, seq SEQ
where GPS.gene_product_id = ASS.gene_product_id
and GPS.seq_id = SEQ.id
and ASS.term_id = GPH.term2_id) as fltPorcArginine,

    (select (sum((SELECT
LENGTH(SEQ.seq)-LENGTH(REPLACE(SEQ.seq,'N',''))))/sum(SEQ.seq_len))
```

```
from gene_product_seq GPS, association ASS, seq SEQ
where GPS.gene_product_id = ASS.gene_product_id
and GPS.seq_id = SEQ.id
and ASS.term_id = GPH.term2_id) as fltPorcAsparagine,

    (select (sum((SELECT
LENGTH(SEQ.seq)-LENGTH(REPLACE(SEQ.seq,'D',''))))/sum(SEQ.seq_len))
from gene_product_seq GPS, association ASS, seq SEQ
where GPS.gene_product_id = ASS.gene_product_id
and GPS.seq_id = SEQ.id
and ASS.term_id = GPH.term2_id) as fltPorcAsparticAcid,

    (select (sum((SELECT
LENGTH(SEQ.seq)-LENGTH(REPLACE(SEQ.seq,'C',''))))/sum(SEQ.seq_len))
from gene_product_seq GPS, association ASS, seq SEQ
where GPS.gene_product_id = ASS.gene_product_id
and GPS.seq_id = SEQ.id
and ASS.term_id = GPH.term2_id) as fltPorcCysteine,

    (select (sum((SELECT
LENGTH(SEQ.seq)-LENGTH(REPLACE(SEQ.seq,'E',''))))/sum(SEQ.seq_len))
from gene_product_seq GPS, association ASS, seq SEQ
where GPS.gene_product_id = ASS.gene_product_id
and GPS.seq_id = SEQ.id
and ASS.term_id = GPH.term2_id) as fltPorcGlutamicAcid,

    (select (sum((SELECT
LENGTH(SEQ.seq)-LENGTH(REPLACE(SEQ.seq,'Q',''))))/sum(SEQ.seq_len))
from gene_product_seq GPS, association ASS, seq SEQ
where GPS.gene_product_id = ASS.gene_product_id
and GPS.seq_id = SEQ.id
and ASS.term_id = GPH.term2_id) as fltPorcGlutamine,

    (select (sum((SELECT
LENGTH(SEQ.seq)-LENGTH(REPLACE(SEQ.seq,'G',''))))/sum(SEQ.seq_len))
from gene_product_seq GPS, association ASS, seq SEQ
where GPS.gene_product_id = ASS.gene_product_id
and GPS.seq_id = SEQ.id
and ASS.term_id = GPH.term2_id) as fltPorcGlycine,
```

```
(select (sum((SELECT
LENGTH(SEQ.seq)-LENGTH(REPLACE(SEQ.seq,'H',''))))/sum(SEQ.seq_len))
from gene_product_seq GPS, association ASS, seq SEQ
where GPS.gene_product_id = ASS.gene_product_id
and GPS.seq_id = SEQ.id
and ASS.term_id = GPH.term2_id) as fltPorcHistidine,

(select (sum((SELECT
LENGTH(SEQ.seq)-LENGTH(REPLACE(SEQ.seq,'I',''))))/sum(SEQ.seq_len))
from gene_product_seq GPS, association ASS, seq SEQ
where GPS.gene_product_id = ASS.gene_product_id
and GPS.seq_id = SEQ.id
and ASS.term_id = GPH.term2_id) as fltPorcIsoleucine,

(select (sum((SELECT
LENGTH(SEQ.seq)-LENGTH(REPLACE(SEQ.seq,'L',''))))/sum(SEQ.seq_len))
from gene_product_seq GPS, association ASS, seq SEQ
where GPS.gene_product_id = ASS.gene_product_id
and GPS.seq_id = SEQ.id
and ASS.term_id = GPH.term2_id) as fltPorcLeucine,

(select (sum((SELECT
LENGTH(SEQ.seq)-LENGTH(REPLACE(SEQ.seq,'K',''))))/sum(SEQ.seq_len))
from gene_product_seq GPS, association ASS, seq SEQ
where GPS.gene_product_id = ASS.gene_product_id
and GPS.seq_id = SEQ.id
and ASS.term_id = GPH.term2_id) as fltPorcLysine,

(select (sum((SELECT
LENGTH(SEQ.seq)-LENGTH(REPLACE(SEQ.seq,'M',''))))/sum(SEQ.seq_len))
from gene_product_seq GPS, association ASS, seq SEQ
where GPS.gene_product_id = ASS.gene_product_id
and GPS.seq_id = SEQ.id
and ASS.term_id = GPH.term2_id) as fltPorcMethionine,

(select (sum((SELECT
LENGTH(SEQ.seq)-LENGTH(REPLACE(SEQ.seq,'F',''))))/sum(SEQ.seq_len))
from gene_product_seq GPS, association ASS, seq SEQ
where GPS.gene_product_id = ASS.gene_product_id
```

```
and GPS.seq_id = SEQ.id
and ASS.term_id = GPH.term2_id) as fltPorcPhenylalanine,

    (select (sum((SELECT
LENGTH(SEQ.seq)-LENGTH(REPLACE(SEQ.seq,'P',''))))/sum(SEQ.seq_len))
from gene_product_seq GPS, association ASS, seq SEQ
where GPS.gene_product_id = ASS.gene_product_id
and GPS.seq_id = SEQ.id
and ASS.term_id = GPH.term2_id) as fltPorcProline,

    (select (sum((SELECT
LENGTH(SEQ.seq)-LENGTH(REPLACE(SEQ.seq,'S',''))))/sum(SEQ.seq_len))
from gene_product_seq GPS, association ASS, seq SEQ
where GPS.gene_product_id = ASS.gene_product_id
and GPS.seq_id = SEQ.id
and ASS.term_id = GPH.term2_id) as fltPorcSerine,

    (select (sum((SELECT
LENGTH(SEQ.seq)-LENGTH(REPLACE(SEQ.seq,'T',''))))/sum(SEQ.seq_len))
from gene_product_seq GPS, association ASS, seq SEQ
where GPS.gene_product_id = ASS.gene_product_id
and GPS.seq_id = SEQ.id
and ASS.term_id = GPH.term2_id) as fltPorcThreonine,

    (select (sum((SELECT
LENGTH(SEQ.seq)-LENGTH(REPLACE(SEQ.seq,'W',''))))/sum(SEQ.seq_len))
from gene_product_seq GPS, association ASS, seq SEQ
where GPS.gene_product_id = ASS.gene_product_id
and GPS.seq_id = SEQ.id
and ASS.term_id = GPH.term2_id) as fltPorcTryptophan,

    (select (sum((SELECT
LENGTH(SEQ.seq)-LENGTH(REPLACE(SEQ.seq,'Y',''))))/sum(SEQ.seq_len))
from gene_product_seq GPS, association ASS, seq SEQ
where GPS.gene_product_id = ASS.gene_product_id
and GPS.seq_id = SEQ.id
and ASS.term_id = GPH.term2_id) as fltPorcTyrosine ,

    (select (sum((SELECT
```

```
LENGTH(SEQ.seq)-LENGTH(REPLACE(SEQ.seq,'V',''))))/sum(SEQ.seq_len))
from gene_product_seq GPS, association ASS, seq SEQ
where GPS.gene_product_id = ASS.gene_product_id
and GPS.seq_id = SEQ.id
and ASS.term_id = GPH.term2_id) as fltPorcValine,

    (CONCAT('CC-',(select TRM3.name
from term TRM3
where TRM3.id = GPH.term1_id))) as strTRMterm_type

    from term TRM, graph_path GPH
where TRM.id = GPH.term1_id
and TRM.id in (
select GPH2.term2_id
from term TRM2, graph_path GPH2
where TRM2.id = GPH2.term1_id
and TRM2.term_type = 'cellular_component'
and TRM2.name = 'cellular_component'
and GPH2.distance = 1)

    and GPH.term1_id < > GPH.term2_id
having fltPorcValine is not null
```

A.2.3 Subclasse de MF

```
    insert IGNORE into SCGO16032013.tbschema (intTRMid, strTRMname,
intQtdeTermosAdjEhPai, intQtdeTermosAdjEhFilho, intDistFilhoMax,intQtdeFilhos,
fltDistMediaFilhos, fltDistDesvPadFilhos, intDistPaiMax, intQtdePais, fltDistMediaPais,
intNroSpecies, intNroGeneProduct, fltMediaProductCountSpecies, intQtdeSeqs,
fltPorcAlanine, fltPorcArginine, fltPorcAsparagine, fltPorcAsparticAcid, fltPorcCysteine,
fltPorcGlutamicAcid, fltPorcGlutamine, fltPorcGlycine, fltPorcHistidine, fltPorcIsoleucine,
fltPorcLeucine, fltPorcLysine, fltPorcMethionine, fltPorcPhenylalanine, fltPorcProline,
fltPorcSerine, fltPorcThreonine, fltPorcTryptophan, fltPorcTyrosine, fltPorcValine,
strTRMterm_type)

    select distinct GPH.term2_id as intTRMid,

    (select TRM3.name
```

```
from term TRM3
where TRM3.id = GPH.term2_id) as strTRMname,

    (select count(T2T2.term2_id)
from term2term T2T2
where T2T2.term1_id = GPH.term2_id) as intQtdeTermosAdjEhPai,

    (select count(T2T2.term1_id)
from term2term T2T2
where T2T2.term2_id = GPH.term2_id) as intQtdeTermosAdjEhFilho ,

    (select max(GPH2.distance)
from graph_path GPH2
where GPH2.term1_id = GPH.term2_id) as intDistFilhoMax ,

    (select count(GPH2.distance)
from graph_path GPH2
where GPH2.term1_id = GPH.term2_id) as intQtdeFilhos ,

    (select avg(GPH2.distance)
from graph_path GPH2
where GPH2.term1_id = GPH.term2_id) as fltDistMediaFilhos ,

    (select std(GPH2.distance)
from graph_path GPH2
where GPH2.term1_id = GPH.term2_id) as fltDistDesvPadFilhos ,

    (select max(GPH2.distance)
from graph_path GPH2
where GPH2.term2_id = GPH.term2_id) as intDistPaiMax ,

    (select count(GPH2.distance)
from graph_path GPH2
where GPH2.term2_id = GPH.term2_id) as intQtdePais ,

    (select avg(GPH2.distance)
from graph_path GPH2
where GPH2.term2_id = GPH.term2_id) as fltDistMediaPais ,

    (select count(GPC.species_id)
from gene_product_count GPC
where GPC.term_id = GPH.term2_id) as intNroSpecies ,
```

```
    (select sum(GPC.product_count)
from gene_product_count GPC
where GPC.term_id = GPH.term2_id) as intNroGeneProduct,

    (select sum(GPC.product_count)/count(GPC.species_id)
from gene_product_count GPC
where GPC.term_id = GPH.term2_id) as fltMediaProductCountSpecies,

    (select count(SEQ.seq)
from gene_product_seq GPS, association ASS, seq SEQ
where GPS.gene_product_id = ASS.gene_product_id
and GPS.seq_id = SEQ.id
and ASS.term_id = GPH.term2_id) as intQtdeSeqs,

    (select (sum((SELECT
LENGTH(SEQ.seq)-LENGTH(REPLACE(SEQ.seq,'A',''))))/sum(SEQ.seq_len))
from gene_product_seq GPS, association ASS, seq SEQ
where GPS.gene_product_id = ASS.gene_product_id
and GPS.seq_id = SEQ.id
and ASS.term_id = GPH.term2_id) as fltPorcAlanine,

    (select (sum((SELECT
LENGTH(SEQ.seq)-LENGTH(REPLACE(SEQ.seq,'R',''))))/sum(SEQ.seq_len))
from gene_product_seq GPS, association ASS, seq SEQ
where GPS.gene_product_id = ASS.gene_product_id
and GPS.seq_id = SEQ.id
and ASS.term_id = GPH.term2_id) as fltPorcArginine,

    (select (sum((SELECT
LENGTH(SEQ.seq)-LENGTH(REPLACE(SEQ.seq,'N',''))))/sum(SEQ.seq_len))
from gene_product_seq GPS, association ASS, seq SEQ
where GPS.gene_product_id = ASS.gene_product_id
and GPS.seq_id = SEQ.id
and ASS.term_id = GPH.term2_id) as fltPorcAsparagine,

    (select (sum((SELECT
LENGTH(SEQ.seq)-LENGTH(REPLACE(SEQ.seq,'D',''))))/sum(SEQ.seq_len))
from gene_product_seq GPS, association ASS, seq SEQ
where GPS.gene_product_id = ASS.gene_product_id
```

```
and GPS.seq_id = SEQ.id
and ASS.term_id = GPH.term2_id) as fltPorcAsparticAcid,

    (select (sum((SELECT
LENGTH(SEQ.seq)-LENGTH(REPLACE(SEQ.seq,'C',''))))/sum(SEQ.seq_len))
from gene_product_seq GPS, association ASS, seq SEQ
where GPS.gene_product_id = ASS.gene_product_id
and GPS.seq_id = SEQ.id
and ASS.term_id = GPH.term2_id) as fltPorcCysteine,

    (select (sum((SELECT
LENGTH(SEQ.seq)-LENGTH(REPLACE(SEQ.seq,'E',''))))/sum(SEQ.seq_len))
from gene_product_seq GPS, association ASS, seq SEQ
where GPS.gene_product_id = ASS.gene_product_id
and GPS.seq_id = SEQ.id
and ASS.term_id = GPH.term2_id) as fltPorcGlutamicAcid,

    (select (sum((SELECT
LENGTH(SEQ.seq)-LENGTH(REPLACE(SEQ.seq,'Q',''))))/sum(SEQ.seq_len))
from gene_product_seq GPS, association ASS, seq SEQ
where GPS.gene_product_id = ASS.gene_product_id
and GPS.seq_id = SEQ.id
and ASS.term_id = GPH.term2_id) as fltPorcGlutamine,

    (select (sum((SELECT
LENGTH(SEQ.seq)-LENGTH(REPLACE(SEQ.seq,'G',''))))/sum(SEQ.seq_len))
from gene_product_seq GPS, association ASS, seq SEQ
where GPS.gene_product_id = ASS.gene_product_id
and GPS.seq_id = SEQ.id
and ASS.term_id = GPH.term2_id) as fltPorcGlycine,

    (select (sum((SELECT
LENGTH(SEQ.seq)-LENGTH(REPLACE(SEQ.seq,'H',''))))/sum(SEQ.seq_len))
from gene_product_seq GPS, association ASS, seq SEQ
where GPS.gene_product_id = ASS.gene_product_id
and GPS.seq_id = SEQ.id
and ASS.term_id = GPH.term2_id) as fltPorcHistidine,

    (select (sum((SELECT
```

```
LENGTH(SEQ.seq)-LENGTH(REPLACE(SEQ.seq,'I',''))))/sum(SEQ.seq_len))
from gene_product_seq GPS, association ASS, seq SEQ
where GPS.gene_product_id = ASS.gene_product_id
and GPS.seq_id = SEQ.id
and ASS.term_id = GPH.term2_id) as fltPorcIsoleucine,

    (select (sum((SELECT
LENGTH(SEQ.seq)-LENGTH(REPLACE(SEQ.seq,'L',''))))/sum(SEQ.seq_len))
from gene_product_seq GPS, association ASS, seq SEQ
where GPS.gene_product_id = ASS.gene_product_id
and GPS.seq_id = SEQ.id
and ASS.term_id = GPH.term2_id) as fltPorcLeucine,

    (select (sum((SELECT
LENGTH(SEQ.seq)-LENGTH(REPLACE(SEQ.seq,'K',''))))/sum(SEQ.seq_len))
from gene_product_seq GPS, association ASS, seq SEQ
where GPS.gene_product_id = ASS.gene_product_id
and GPS.seq_id = SEQ.id
and ASS.term_id = GPH.term2_id) as fltPorcLysine,

    (select (sum((SELECT
LENGTH(SEQ.seq)-LENGTH(REPLACE(SEQ.seq,'M',''))))/sum(SEQ.seq_len))
from gene_product_seq GPS, association ASS, seq SEQ
where GPS.gene_product_id = ASS.gene_product_id
and GPS.seq_id = SEQ.id
and ASS.term_id = GPH.term2_id) as fltPorcMethionine,

    (select (sum((SELECT
LENGTH(SEQ.seq)-LENGTH(REPLACE(SEQ.seq,'F',''))))/sum(SEQ.seq_len))
from gene_product_seq GPS, association ASS, seq SEQ
where GPS.gene_product_id = ASS.gene_product_id
and GPS.seq_id = SEQ.id
and ASS.term_id = GPH.term2_id) as fltPorcPhenylalanine,

    (select (sum((SELECT
LENGTH(SEQ.seq)-LENGTH(REPLACE(SEQ.seq,'P',''))))/sum(SEQ.seq_len))
from gene_product_seq GPS, association ASS, seq SEQ
where GPS.gene_product_id = ASS.gene_product_id
and GPS.seq_id = SEQ.id
```

```
and ASS.term_id = GPH.term2_id) as fltPorcProline,

    (select (sum((SELECT
LENGTH(SEQ.seq)-LENGTH(REPLACE(SEQ.seq,'S',''))))/sum(SEQ.seq_len))
from gene_product_seq GPS, association ASS, seq SEQ
where GPS.gene_product_id = ASS.gene_product_id
and GPS.seq_id = SEQ.id
and ASS.term_id = GPH.term2_id) as fltPorcSerine,

    (select (sum((SELECT
LENGTH(SEQ.seq)-LENGTH(REPLACE(SEQ.seq,'T',''))))/sum(SEQ.seq_len))
from gene_product_seq GPS, association ASS, seq SEQ
where GPS.gene_product_id = ASS.gene_product_id
and GPS.seq_id = SEQ.id
and ASS.term_id = GPH.term2_id) as fltPorcThreonine,

    (select (sum((SELECT
LENGTH(SEQ.seq)-LENGTH(REPLACE(SEQ.seq,'W',''))))/sum(SEQ.seq_len))
from gene_product_seq GPS, association ASS, seq SEQ
where GPS.gene_product_id = ASS.gene_product_id
and GPS.seq_id = SEQ.id
and ASS.term_id = GPH.term2_id) as fltPorcTryptophan,

    (select (sum((SELECT
LENGTH(SEQ.seq)-LENGTH(REPLACE(SEQ.seq,'Y',''))))/sum(SEQ.seq_len))
from gene_product_seq GPS, association ASS, seq SEQ
where GPS.gene_product_id = ASS.gene_product_id
and GPS.seq_id = SEQ.id
and ASS.term_id = GPH.term2_id) as fltPorcTyrosine,

    (select (sum((SELECT
LENGTH(SEQ.seq)-LENGTH(REPLACE(SEQ.seq,'V',''))))/sum(SEQ.seq_len))
from gene_product_seq GPS, association ASS, seq SEQ
where GPS.gene_product_id = ASS.gene_product_id
and GPS.seq_id = SEQ.id
and ASS.term_id = GPH.term2_id) as fltPorcValine,

    (CONCAT('MF-',(select TRM3.name
from term TRM3
```

```
where TRM3.id = GPH.term1_id))) as strTRMterm_type

    from term TRM, graph_path GPH
where TRM.id = GPH.term1_id
and TRM.id in (
select GPH2.term2_id
from term TRM2, graph_path GPH2
where TRM2.id = GPH2.term1_id
and TRM2.term_type = 'molecular_function'
and TRM2.name = 'molecular_function'
and GPH2.distance = 1)

    and GPH.term1_id <> GPH.term2_id
having fltPorcValine is not null
```

Apendice B

MODELO DE DADOS UTILIZADO PARA A CONSTRUÇÃO DAS CONSULTAS SQL

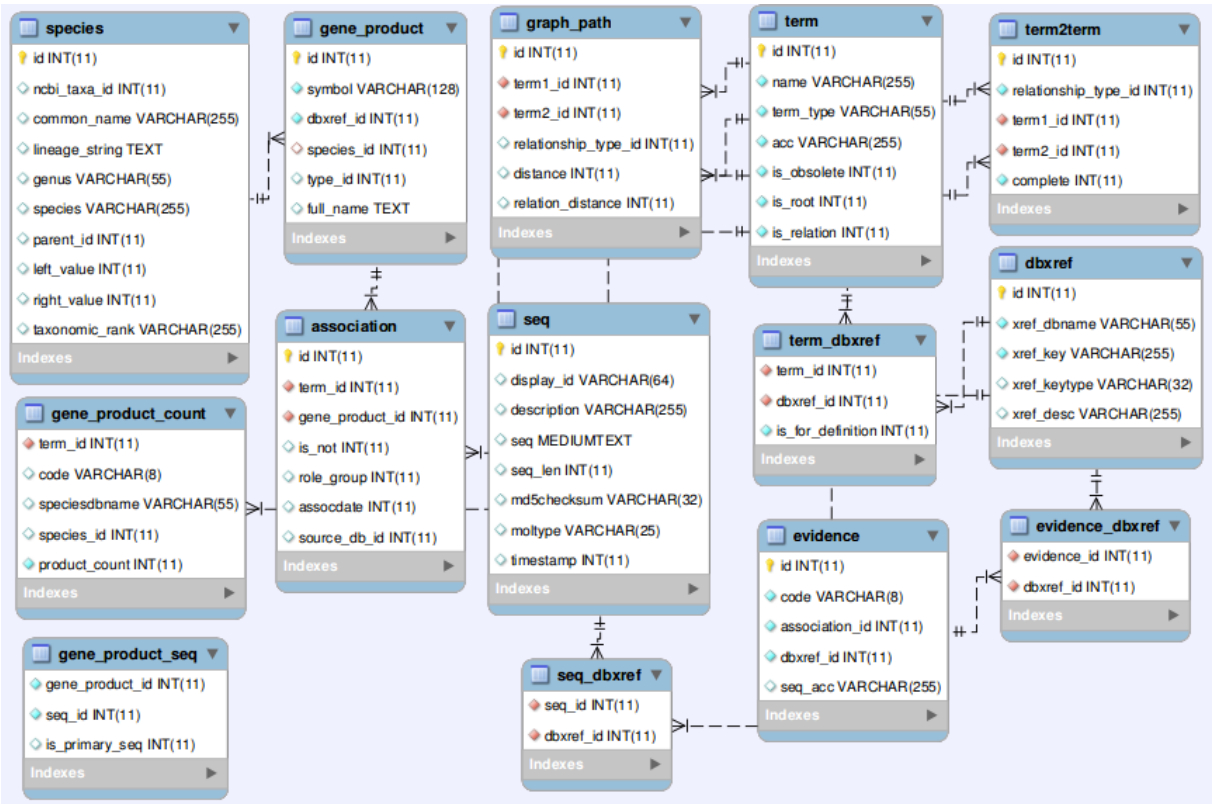

Figura B.1: Modelo de dados utilizado para a construção das consultas SQL

Apendice C

RESULTADOS COMPLETOS - CEE E $C_{1,2,5,6,7}$

Tabela C.1: Resultados encontrados nas 35 execuções para o método CEE e para as configurações $C_{1,2,5,6,7}$

Semente	**CEE**	C_1	C_2	C_5	C_6	C_7
1	74,43	79,10	77,45	83,81	88,53	86,75
2	68,28	78,20	77,28	83,83	87,59	85,72
3	70,98	77,49	78,05	84,03	87,69	87,14
4	74,02	79,09	76,52	83,86	87,90	86,26
5	70,36	79,36	77,80	83,70	88,16	87,24
6	72,15	78,46	76,99	83,93	90,19	85,49
7	69,87	78,19	77,74	83,61	87,69	85,42
8	72,57	78,99	76,73	83,90	88,18	87,05
9	73,98	78,24	76,89	83,58	87,87	86,45
10	75,11	79,36	76,77	83,61	90,29	86,10
11	74,14	79,61	77,40	83,56	87,93	84,91
12	71,22	79,38	75,55	83,63	89,64	86,26
13	70,18	78,95	78,49	84,09	89,74	86,86
14	67,62	80,55	77,39	83,81	90,26	86,33
15	74,71	78,36	77,50	84,05	88,09	86,62
16	74,13	78,87	78,16	84,12	88,11	89,07
17	74,38	79,20	77,74	83,92	87,81	88,95
18	74,05	78,20	78,07	84,03	87,75	86,32
19	74,35	78,03	75,84	83,61	87,55	86,01
20	73,90	78,19	78,98	84,43	88,63	86,77
21	72,36	79,10	78,34	83,61	87,82	86,24
22	73,63	79,08	77,38	83,74	88,30	87,06
23	72,65	79,06	77,36	83,61	90,35	86,04
24	70,23	79,20	77,90	83,63	89,90	85,12
25	64,44	78,12	77,10	83,71	88,11	86,43
26	73,96	79,36	76,98	84,29	88,63	85,78
27	78,71	80,40	78,56	84,39	90,00	87,25
28	72,56	80,03	78,49	83,57	88,11	84,56
29	69,61	80,63	77,90	83,55	87,91	87,22
30	71,55	80,62	77,42	84,12	89,97	86,29
31	72,40	79,02	77,10	84,09	88,18	86,37
32	74,24	80,61	77,73	83,97	90,00	85,64
33	73,96	79,10	75,40	83,77	88,22	85,01
34	70,47	78,19	76,03	84,09	88,32	84,79
35	71,68	80,58	77,38	83,66	88,57	87,08
Média	72,37	79,11	77,38	83,85	88,63	86,36
Desvio Padrão	2,60	0,86	0,83	0,25	0,95	0,99
Intervalo de Confiança	0,86	0,28	0,28	0,08	0,31	0,33
Valor mínimo	71,53	78,8	77,13	83,76	88,3	86,0
Valor máximo	73,19	79,42	77,63	83,94	88,94	86,7