Departamento de Ciência da Computação
Instituto de Matemática e Estatística
Universidade de São Paulo

Trabalho de Formatura Supervisionado

Implementação de um sistema de validação estatística configurável de dados

Eduardo Dias Filho

Orientadores:
Prof. Dr. João Eduardo Ferreira
Dr. Pedro Losco Takecian

2014

Resumo

Filho, E. D. **Implementação de um sistema de validação estatística configurável de dados**. Monografia - Instituto de Matemática e Estatística, Universidade de São Paulo, São Paulo, 2014.

O objetivo deste trabalho é implementar um sistema que seja capaz de aplicar a validação estatística para lotes de dados através da técnica de detecção de anomalias.

Além disso, o sistema deve oferecer uma interface que permita configurar a validação estatística adaptando-a a um problema específico e que também permita administrar o treinamento dos detectores de anomalias.

Sumário

Parte I

Parte Objetiva

1 Introdução

1.1 Motivação

Com a popularização de sistemas que coletam grandes volumes de dados de fontes diversas, mostrou-se necessário garantir a integração e corretude da informação recebida. Neste trabalho, tais volumes de dados estão dispostos em forma de lotes, que são conjuntos de instâncias de uma ou mais estrutura de dados.

Geralmente lotes são representados por um arquivo comprimido que contém um arquivo para cada uma de suas estruturas de dados. Estes arquivos são arquivos de valores separados por vírgula (CSV - Comma Separated Values) nos quais as linhas representam as instâncias que compõem o lote, ou seja, cada linha é uma sequência de valores dos atributos especificados na estrutura.

As tabelas a seguir ilustram como um lote é composto. Na tabela 1 o lote é representado pelo arquivo "Lote1.zip" e os arquivos CSV que o compõem são listados. Na tabela 2 é exibido o conteúdo do arquivo "Pessoas.csv", suas linhas representam instâncias da estrutura de dados representada pelo arquivo, na primeira linha da tabela 2 são descritos os atributos de tal estrutura.

Tabela 1: Exemplo de lote

Arquivos de "Lote1.zip"
Pessoas.csv
Produtos.csv
...

Tabela 2: Exemplo do conteúdo do arquivo "Pessoas.csv" do lote "Lote1.zip"

Pessoas.csv		
Nº da linha	Nome	Idade
1	Ana	29
2	José	32
3	Maria	44
...	...	...

Para alcançar a integração e corretude dos lotes, as estruturas do lote recebido devem estar organizados em um formato preestabelecido e cada um de seus atributos deve respeitar um determinado domínio, ou seja, o sistema deve realizar uma validação sintática para checar estas condições. Também é necessário que os dados estejam de acordo com um conjunto de regras lógicas predeterminadas, isto é, o sistema deve realizar uma validação semântica. Porém as medidas citadas são insuficientes para garantir a corretude, pois embora o lote siga as regras sintáticas e semânticas, ainda pode não representar a realidade por conta de algum possível erro de coleta.

Pelo fato de um lote possuir muitas instâncias, um lote que contenha um erro deste tipo pode ser detectado caso, devido ao erro, as porcentagens de ocorrência dos valores de um atributo no lote diferirem o suficiente das porcentagens de ocorrência dos dos valores do mesmo atributo em outros lotes considerados corretos.

Assim, a partir de uma amostra de lotes previamente recebidos e classificados em válidos ou inválidos por um especialista de domínio, é possível usar uma técnica chamada de detecção de anomalias para decidir se as informações de um novo lote recebido estão provavelmente corretas ou incorretas. Esta etapa de validação é chamada de validação estatística.

1.2 Contexto

Inicialmente, o objetivo deste trabalho era implementar a validação estatística para o sistema de integração de dados Bloddis (Blood donation data integration system - disponível em https://data.ime.usp.br/bloddis), que recebe lotes de diversos hemocentros.

A validação estatística para os lotes do Bloddis consiste em uma série de detecções de anomalias sobre os atributos dos lotes selecionados para validação. Devido à complexidade da validação estatística, surgiu a necessidade de uma interface de gerenciamento para facilitar futuras alterações no processo de validação e realizar o controle dos conjuntos de treinamento.

Portanto, este trabalho visa implementar um método de validação estatística que faz uso de algoritmo de detecção de anomalias e desenvolver uma interface para administrar a configuração e funcionamento do processo de validação estatística. Deste modo, através da especificação da estrutura dos lotes enviados e de quais atributos serão usados para decidir se o lote é anômalo, a interface permite ao usuário adaptar a validação estatística de dados para um domínio desejado.

2 Fundamentos

2.1 Detecção de anomalias

Detecção de anomalias refere-se ao problema de encontrar exemplares de dados que não seguem o comportamento esperado em uma amostra. Tais exemplares são geralmente chamados de anomalias [CBK09].

Existem diversos métodos para realizar a detecção de anomalias. Neste trabalho serão abordados dois métodos que mostraram melhores resultados para detectar anomalias nos lotes do Bloddis, a detecção gaussiana e a sua variante detecção gaussiana multivariada, estudados em [Ng14].

Ambos os métodos visam rotular um lote como anômalo ou não-anômalo a partir de uma amostra A de lotes já classificados por um especialista do domínio ou pelo próprio algoritmo de detecção, ou seja, são métodos de aprendizado computacional semi-supervisionado pois fazem uso de classificações feitas pelo próprio algoritmo para melhorar seu desempenho.

Uma detecção de anomalia leva em conta apenas um atributo do lote para classificá-lo, tal atributo é chamado de atributo comportamental. Para classificar o lote com base no atributo comportamental é preciso medir as porcentagens de ocorrências de seus possíveis valores nas instâncias do lote para compará-las com as porcentagens dos outros lotes já classificados.

Assim, para medir os valores do atributo comportamental são usadas unidades de medida que podem ser categorias, intervalos ou grupos. Para ilustrar tais conceitos, suponha que um lote de dados de doações sanguíneas contém um arquivo de doadores e que há o interesse em usar o atributo "Grupo ABO", que informa o grupo sanguíneo do doador, como atributo comportamental. Neste caso, há poucos possíveis valores que as instâncias podem ter atribuídos ao atributo comportamental, são eles "O", "A", "B" e "AB". Assim, pode-se usar os próprios valores como categorias, mas se houver interesse pode-se também usar grupos de categorias para medir os possíveis valores, como por exemplo "AB" e "Não AB". Suponha agora que o atributo comportamental usado é o atributo "Idade", que informa a idade do doador. Neste caso, como há muitos possíveis valores, pode-se usar intervalos de valores para medir a ocorrência, por exemplo, é possível ter os seguintes intervalos "0 a 9 anos", "10 a 19 anos", "20 a 29 anos", "30 a 39 anos", "40 a 49 anos", "50 a 59 anos", "60 a 69 anos", "70 anos ou mais".

Nas próximas seções serão exibidos os métodos usados para realizar essa comparação entre as porcentagens de ocorrência das categorias, grupos ou intervalos de um lote enviado com as porcentagens dos lotes já classificados e chegar a uma classificação para o lote enviado. Para facilitar a notação, os lotes serão representados como tuplas da forma $X = (X_1, ..., X_n)$, aonde cada X_i representa a porcentagem de ocorrência de uma categoria, grupo ou intervalo do atributo comportamental no lote.

Por exemplo, suponha que o arquivo de doadores de um lote de doações sanguíneas L1 possui 10000 instâncias e a distribuição das ocorrências das categorias nas instâncias em L1 são descritas na tabela 3. Assim, o lote L1 é representado pela tupla X = (0.485, 0.365, 0.115, 0.035).

Tabela 3: Quantidade das ocorrências das possíveis categorias do atributo "grupo ABO" nas instâncias do lote L1

Grupo ABO	Quantidade de instâncias
O	4850
A	3650
B	1150
AB	350

2.2 Detecção gaussiana

O método da detecção gaussiana assume que as porcentagens de ocorrência das categorias, grupos ou intervalos do atributo comportamental de um lote seguem uma função de distribuição de probabilidade normal. Assim, a partir de uma amostra de lotes já classificados como não-anômalos, é possivel deduzir a média μ_i e a variância σ_i de cada porcentagem de ocorrência X_i de uma categoria, grupo ou intervalo na amostra.

Deste modo, a função de densidade de probabilidade $P(X_i, \mu_i, \sigma_i)$ para cada porcentagem de ocorrência X_i de uma categoria em um lote $X = (X_1, ..., X_n)$ é dada por:

$$P(X_i, \mu_i, \sigma_i) = \frac{1}{\sigma_i\sqrt{2\pi}} e^{-(X_i-\mu_i)^2/2\sigma_i^2}$$

É importante observar que o conjunto de lotes usados nos cálculos de μ_i e σ_i, também chamado de conjunto de treinamento, deve ser constituído por lotes classificados como não-anômalos, já que estes definem o padrão que será comparado com o lote a ser classificado.

Além de integrarem o conjunto de treinamento, lotes já classificados também serão usados em outra etapa da detecção gaussiana, a etapa de validação, que será introduzida ao longo da seção. Deste modo, os lotes já classificados são separados em dois conjuntos disjuntos, o conjunto de treinamento usado para calcular μ_i e σ_i e o conjunto de validação. Por convenção, o conjunto de treinamento é composto por cerca de 60% dos lotes já classificados, o restante dos lotes farão parte do conjunto de validação.

A detecção gaussiana também assume que as porcentagens de ocorrência das categorias do atributo comportamental são independentes entre si. Assim, a função densidade de probabilidade conjunta das variáveis da tupla $X = (X_1, ..., X_n)$ é dada por:

$$P(X) = \prod_{i=1}^{n} P(X_i, \mu_i, \sigma_i)$$

Assim, caso P(X) seja suficientemente pequeno, pode-se dizer que a configuração das porcentagens de ocorrência do lote X foge do padrão e portanto X é considerado uma anomalia. De outro modo:

Se $P(X) < \varepsilon$, então X é classificado como anomalia.
Se $P(X) >= \varepsilon$, então X é classificado como normal.

A próxima etapa do algoritmo visa decidir o valor do parâmetro ε a ser usado para comparar com P(X) tal que o algoritmo tenha o melhor desempenho possível na classificação de lotes. Caso ε seja muito pequeno, o algoritmo poderá classificar lotes anômalos como não-anômalos e caso ε seja muito grande, o algoritmo poderá classificar um lote não-anômalo como anomalia. Para determinar um valor adequado para ε é realizada a etapa de validação.

2.2.1 Etapa de validação

A etapa de validação consiste em calcular os valores da função densidade de probabilidade conjunta P(X) para cada lote X do conjunto de validação e, para cada ε possível, comparar os valores de P(X) com ε para gerar uma nova classificação para cada lote X, que será comparada à classificação inicial do lote X para assim encontrar o ε que maximiza o desempenho do algoritmo sobre os lotes do conjunto de validação.

Deste modo, é necessário iterar pelos possíveis valores de ε para testar o desempenho do algoritmo para cada um deles. Portanto deve-se definir um intervalo de valores elegíveis de ε.

Seja P o conjunto dos valores da função densidade de probabilidade conjunta P(X) calculados a partir dos lotes X do conjunto de validação e sejam min{P} e max{P} respectivamente o menor e o maior dos valores do conjunto P, é importante observar que os possíveis valores de ε estão contidos no intervalo]min{P}, max{P}]. Pois para todos os ε menores ou iguais a min{P}, todos os lotes do conjunto de validação serão classificados como não-anômalos já que para todo X no conjunto de validação $P(X) >= min\{P\} >= \varepsilon$, então é desnecessário testar o algoritmo para valores de ε menores ou iguais a min{P}. Da mesma forma, para qualquer ε maior que max{P}, todos os lotes do conjunto de validação serão classificados como anomalias já que para todo X no conjunto de validação $P(X) <= max\{P\} < \varepsilon$, portanto basta testar até $\varepsilon = max\{P\}$. Deste modo, escolhendo uma quantidade de valores a serem testados, por exemplo, 1000, o desempenho do algoritmo é medido para 1000 valores no intervalo]min{P}, max{P}].

Para medir o desempenho do algoritmo para um certo ε geralmente é usada uma medida chamada F1-Score, calculada a partir da comparação das classificações originais dos lotes do conjunto de validação com as classificações novas obtidas usando um valor específico ε. Sejam:

- VP o número de verdadeiros positivos, ou seja, lotes classificados como "anômalo", cuja classificação original era "anômalo".
- FP o número de falsos positivos, ou seja, lotes classificados como "anômalo", cuja classificação original era "não-anômalo".
- FN o número de falsos negativos, ou seja, lotes classificados como "não-anômalo", cuja classificação original era "anômalo".
- P a precisão dada por $P = \frac{VP}{VP+FP}$.
- C a cobertura dada por $C = \frac{VP}{VP+FN}$.

Então, temos:

$$\text{F1-Score} = \frac{2*P*C}{P+C}$$

Deste modo, o maior valor de F1-Score é 1 e o menor é 0, o objetivo desta etapa é encontrar um valor de ε que maximize o F1-Score. Após esta etapa o método pode usar ε para comparar com P(X) e classificar o lote representado pela tupla X como anomalia ou não-anômalo.

2.3 Detecção gaussiana multivariada

O método da detecção gaussiana mostrado na seção anterior assume a independência entre variáveis, entretanto existe uma variante do método, a detecção gaussiana multivariada, que leva em conta a dependência entre variáveis. Devido à dependência das probabilidades de ocorrência das características, a função de densidade de probabilidade conjunta do lote X é calculada de outra forma.

Seja $\mu \in \mathbb{R}^m$ o vetor no qual μ_i é a média da variável X_i dos lotes do conjunto de treinamento e seja Σ a matriz de covariância, calculada por $\Sigma_{i,j} = COV(X_i, X_j)$, onde $COV(X_i, X_j)$ é a covariância entre as variáveis X_i e X_j dos lotes do conjunto de treinamento. Temos:

$$P(X) = \frac{1}{(2\pi)^{\frac{n}{2}}|\Sigma|^{\frac{1}{2}}} e^{(-\frac{1}{2}(X-\mu)^T\Sigma^{-1}(X-\mu))}.$$

Assim como no método anterior, a detecção multivariada usa P(X) para comparar com um ε determinado pela etapa de validação e decidir se o lote X é ou não anômalo.

3 Tecnologias

3.1 *Interface* de configuração e treinamento

A *interface* de configuração e treinamento de detectores de anomalias foi implementada como uma aplicação *web* usando o *framework* Django para desenvolvimento *web* com a linguagem Python. Para armazenar os dados de validação e poder acessá-los pela *interface*, foi criado um banco de dados usando o sistema gerenciador de banco de dados PostgreSQL. Nesta seção serão apresentados maiores detalhes sobre tais tecnologias e como foram usadas na implementação da *interface* de configuração e treinamento.

3.1.1 Django

Django é um *framework web* de alto nível para Python que encoraja o desenvolvimento rápido e a concepção pragmática e limpa [Fou14a].

A metodologia MVC (*Model-view-controller*) consiste em separar a aplicação em três camadas, uma contendo as partes que representam o modelo do domínio da aplicação (*Model*), uma que contém o modo como o modelo é apresentado ao usuário (*View*) e uma que informa como o usuário interage com a aplicação (*Controller*)[KP88].

A arquitetura do Django se baseia em uma variação de MVC na qual a camada de *view* descreve quais dados serão apresentados ao usuário ao invés do modo como serão apresentados ao usuário, que está descrito em uma camada chamada *template*. Além disso, no Django a camada de *controller* está contida no próprio *framework* que dependendo da entrada do usuário e da configuração de URLs do Django, faz a requisição de uma *view*. Deste modo, geralmente o desenvolvimento de uma aplicação em Django está concentrado nas camadas de *model*, *template* e *view* e portanto é comum chamar a arquitetura do Django de MTV(*Model-template-view*) , embora não deixe de ser um *framework* MVC[Fou14b].

A camada *model* do Django permite ao desenvolvedor definir as classes de modelagem que representam as tabelas do banco de dados da aplicação. Para relacionar o banco de dados com as classes de modelagem, o Django permite que o desenvolvedor realize suas próprias consultas SQL, mas também oferece métodos nas classes de modelagem que facilitam a leitura e escrita no banco de dados. Para que o Django se conecte com o banco de dados, o desenvolvedor deve configurar o arquivo "settings.py", na pasta da aplicação, com os dados do servidor do banco de dados.

No Django a camada de *view* consiste em um conjunto de funções definidas pelo desenvolvedor que recebem uma requisição HTTP e retornam uma resposta HTTP com os dados a serem exibidos para o usuário. Para controlar qual destas funções de *view* é acionada, o desenvolvedor deve criar uma configuração de URL, que mapeia um padrão de URL a uma função de *view* que retornará a resposta quando o endereço for requisitado.

A apresentação dos dados retornados na resposta de uma função de *view* é a responsabilidade da camada de *template*. A apresentação dos dados é descrita em um arquivo de *template*, que é um simples arquivo de texto que gera um arquivo baseado em texto como HTML, XML ou CSV. O arquivo de *template* deve ser escrito pelo desenvolvedor na linguagem de *template* do Django, que possibilita o uso de variáveis e outros operadores que facilitam a montagem da apresentação dos dados.

Para a interface de configuração, os arquivos HTML gerados pelos arquivos de *template* fazem uso da biblioteca jQuery de desenvolvimento JavaScript para prover dinamismo à apresentação dos dados. Além disso, também é usado código CSS para definir estilos para os elementos do arquivo HTML.

3.1.2 PostgreSQL

Os dados das configurações dos processos de validação e de lotes usados no treinamento dos validadores são armazenados em um banco de dados usando o sistema gerenciador de banco de dados PostgreSQL.

PostgreSQL é um poderoso sistema de banco de dados objeto-relacional *open source*. Tem mais de 15 anos de desenvolvimento ativo e uma arquitetura que ganhou uma forte reputação por conta de sua confiabilidade, integridade dos dados e corretude [Gro14].

3.2 Algoritmos de detecção de anomalias

Os algoritmos de detecção gaussiana e detecção gaussiana multivariada foram implementados em módulos da linguagem Python, os módulos ficam no servidor da interface e são executados quando é requisitada a validação de um lote através de algum dos métodos.

Para implementar o cálculo estatístico necessário para a detecção de anomalias são usadas as bibliotecas Numpy e Scipy do Python.

3.2.1 Numpy e Scipy

Numpy é o pacote fundamental para computação científica com Python [dev13]. Os componentes do Numpy mais usados neste trabalho foram os objetos de *array* N-dimensionais usados para armazenar dados dos lotes de treinamento, assim como funções para manipular tais objetos.

Scipy é uma coleção de algoritmos matemáticos e funções de conveniência construídos sobre Numpy [com14]. O Scipy é organizado em módulos que oferecem funções para as diversas áreas da ciência, para calcular as funções de densidade de probabilidade foram usadas funções do módulo estatístico do Scipy.

4 Implementação

Nesta seção será apresentada a implementação do método de validação estatística de lotes proposto em [Tak09]. Para melhor compreensão do método de validação aqui implementado e de suas estruturas é recomendada a leitura do capítulo 5 de [Tak09], este trabalho trata apenas da implementação deste método.

No método proposto, chama-se processo de validação a estrutura que recebe como entrada um lote e retorna a classificação do lote entre "válido" ou "inválido". E chama-se validador a estrutura que classifica um lote baseando-se em apenas um atributo de interesse, chamado de atributo comportamental.

Deste modo, um validador não precisa receber o lote inteiro, mas apenas uma versão reduzida do lote, chamada de lote reduzido. Um lote reduzido é composto com os dados do lote original que são de interesse do validador, ou seja, o valor do atributo comportamental de cada instância e o valor do atributo usado como chave primária que identifica cada instância unicamente. Assim, um validador recebe um lote reduzido e retorna sua classificação entre "válido" ou "inválido". Para realizar a classificação, o validador usa um algoritmo detector de anomalias enviando as porcentagens de ocorrência de cada categoria ou intervalo do atributo comportamental no lote reduzido.

Um processo de validação contém um ou mais validadores. Então quando um lote é submetido ao processo de validação, sua classificação é dada pela conjunção das classificações retornadas por cada validador, ou seja, se todos os validadores de um processo de validação retornam "válido", o processo também retorna "válido", caso um dos validadores retorne "inválido", então o processo de validação também retorna "inválido".

Suponha que em um validador, as porcentagens de ocorrência das categorias do atributo comportamental dependam de alguma forma dos valores de um ou mais atributos do lote, que serão chamados aqui de atributos contextuais e cada diferente conjunto de categorias dos atributos contextuais é chamado de contexto. Assim, para cada diferente contexto pode existir um diferente padrão de distribuição das categorias do atributo comportamental. Isso dificulta a detecção de anomalia pois uma ocorrência que seria considerada anômala em um dos padrões, pode ser considerada normal quando consideramos todos os padrões na mesma detecção.

Para resolver este problema de dependência entre atributos, os validadores são divididos em dois tipos:

Validadores lógicos que são estruturas responsáveis por classificar um lote reduzido baseando-se em um atributo comportamental e levando em conta a possível existência de contextos. A especificação de um validador lógico deve conter os atributos contextuais se estes existirem. Um validador lógico não está diretamente associado a um algoritmo detector de anomalias.

Validadores físicos que são estruturas responsáveis por classificar um lote reduzido baseando-se em um atributo comportamental em um determinado contexto. O lote reduzido recebido por um validador físico contém os valores do atributo comportamental e do atributo usado como chave primária que identifica cada instância, mas contém apenas as instâncias que fazem parte do contexto do validador físico. Um validador físico está associado a um algoritmo detector de anomalias para classificar o lote reduzido que recebe.

Um validador lógico contém um ou mais validadores físicos e classifica o lote reduzido que recebe usando a conjunção das classificações de seus validadores físicos, ou seja, se todos

os validadores físicos de um validador lógico retornam "válido", o validador lógico também retorna "válido", caso um dos validadores físicos retorne "inválido", então o validador lógico também retorna "inválido" . E por fim, a classificação retornada pelo processo de validação é uma conjunção das classificações dos validadores lógicos que o compõem.

Neste método, é sugerida a criação de um núcleo para conter as implementações dos algoritmos de detecção de anomalias a serem usados. No núcleo, cada algoritmo é implementado em um módulo isolado e todos os módulos seguem a interface sugerida pelo método de validação estatística proposto, ou seja, todos os algoritmos de detecção interagem com o resto do sistema através de um mesmo conjunto de métodos.

Além disso, o método sugere a criação de uma interface web que possibilita ao usuário gerenciar a validação estatística através de três principais funções:

- **Configuração da validação estatística**: Permite configurar um processo de validação, ou seja, definir especificações para adaptar a validação estatística ao problema desejado.
- **Administração dos lotes de treinamento**: Permite enviar e gerenciar os lotes usados para treinar os algoritmos detectores de anomalias de um processo de validação.
- **Validação**: Permite o envio de lotes que serão submetidos ao processo de validação escolhido.

Assim, quando um usuário configura um processo de validação para resolver um problema específico, estas informações são armazenadas em um banco de dados e os validadores físicos que compõem os validadores lógicos do processo de validação em questão instanciam um objeto detector de anomalias no núcleo. Sempre que um lote de treinamento é enviado ou alterado, as informações são escritas no banco de dados e nos objetos detectores de anomalias relacionados com a operação. Quando o usuário submete um arquivo ao processo de validação criado, os objetos detectores relacionados aos validadores físicos que fazem parte do processo de validação devem ser usados para classificar o lote. Além disso, um lote enviado para validação também pode ser usado como lote de treinamento para validações futuras.

Nas subseções a seguir será exibida a implementação das estruturas de configuração, controle de lotes de treinamento e validação de lotes no banco de dados e na interface. Além disso, também será mostrada a implementação de um módulo de detecção de anomalias do núcleo baseado no algoritmo de detecção gaussiana visto na seção 2.

4.1 Configuração da validação estatística

A configuração da validação estatística permite ao usuário especificar como deve ser realizada a validação estatística para seu problema através da interface, que oferece telas de criação e gerenciamento das estruturas de configuração do método proposto.

Segundo o método de validação proposto, para configurar um processo de validação pelo qual o lote será submetido é necessário dar-lhe um nome, descrever a estrutura dos lotes esperados pelo processo e especificar os validadores lógicos que compõem o processo de validação.

Esta subseção mostrará como tais estruturas estão organizadas no banco de dados e como é realizado o gerenciamento das estruturas de configuração através da interface.

4.1.1 Estrutura de lote

Uma estrutura de lotes especifica como os dados de um lote estão dispostos. Como descrito na subseção 1.1, um lote é composto de arquivos que representam estruturas de dados. As informações de tais arquivos estão organizadas em entidades chamadas de estruturas de arquivos, que contém o nome do arquivo e os dados de seus atributos. Assim, a estrutura de lotes é composta por estruturas de arquivos.

Além disso, uma estrutura de lotes tem um nome e tem também um campo de estado que indica se a estrutura está em uso por algum processo de validação ou se a estrutura é nova e não está em uso. O estado de uma estrutura de lote é importante para que o sistema possa controlar quais operações devem ser permitidas ao usuário de modo a preservar a consistência da validação.

Se a estrutura estiver com o estado "nova" ela pode ser alterada e removida através da interface com o usuário, porém se o estado for "em uso", uma alteração ou remoção causaria erros em todos os processos de validação que usam a estrutura de lote, portanto, neste caso não são permitidas alterações ou remoções.

Independentemente do estado de uma estrutura de lote, a interface permite ao usuário realizar uma duplicação da estrutura de lote, originando uma nova estrutura com a mesma especificação da estrutura duplicada, mas com o estado "nova", ou seja, uma cópia que não está em uso. Outra operação que também é sempre permitida ao usuário é a criação de uma nova estrutura de lotes vazia com o nome dado pelo usuário e estado "nova".

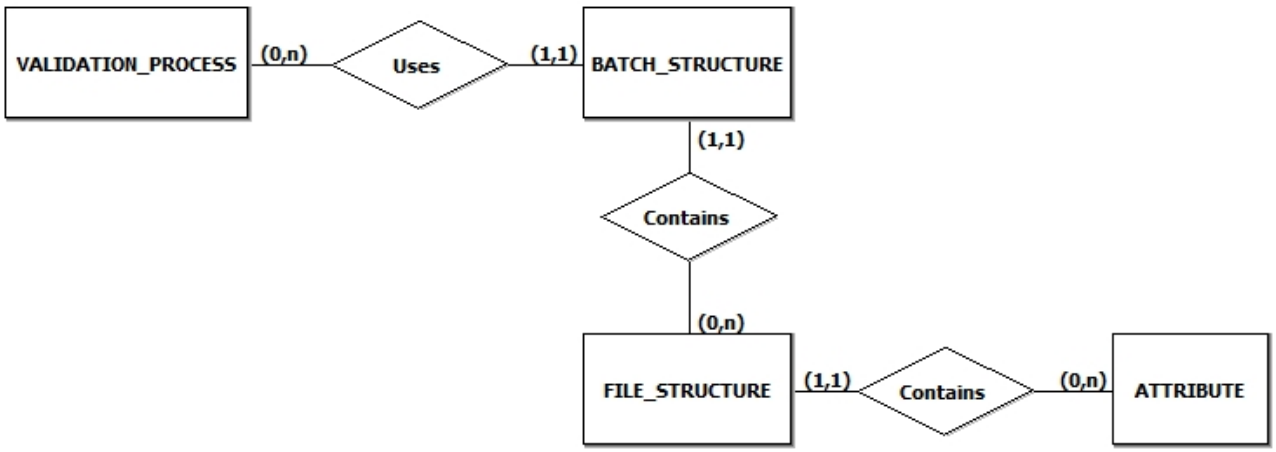


Figura 1: Diagrama da modelagem conceitual do banco de dados para estrutura de lotes

Na modelagem conceitual do banco de dados há uma entidade que representa a estrutura de lote ("BATCH_STRUCTURE") relacionada com a entidade que representa o processo de validação ("VALIDATION_PROCESS") através de um relacionamento de uso, o que significa que o processo de validação usa apenas uma estrutura de lote, embora a estrutura de lote possa ser usada por zero ou mais processos de validação.

Além disso, uma estrutura de lote também está relacionada a entidade estrutura de arquivo ("FILE_STRUCTURE") que, por sua vez, está relacionada a entidade que armazena as informações de cada atributo ("ATTRIBUTE"). Os dois últimos relacionamentos são da forma "contém", isso significa que uma estrutura de lotes contém uma estrutura de arquivos e uma estrutura de arquivos contém atributos.

Todos os relacionamentos da figura 1 são feitos a partir de chave estrangeira, no relacionamento de uso entre processo de validação e estrutura de lotes, o processo guarda a chave da estrutura de lote que usa. Da mesma forma, no relacionamento "contém" entre estrutura de lotes e estrutura de arquivos, a estrutura de arquivo guarda a chave da estrutura de lote que a contém. Por fim, no relacionamento "contém" entre estrutura de arquivos e atributo, o atributo guarda a chave da estrutura de arquivos que a contém.

Para mostrar como o usuário interage com a interface durante a criação e configuração de uma estrutura de lotes, tome como exemplo um lote com dados de doações sanguíneas. O usuário precisa definir a estrutura de lote que será usada por um processo de validação. Assim, através da interface, o usuário deve criar uma nova estrutura de lote como mostrado na figura 2.

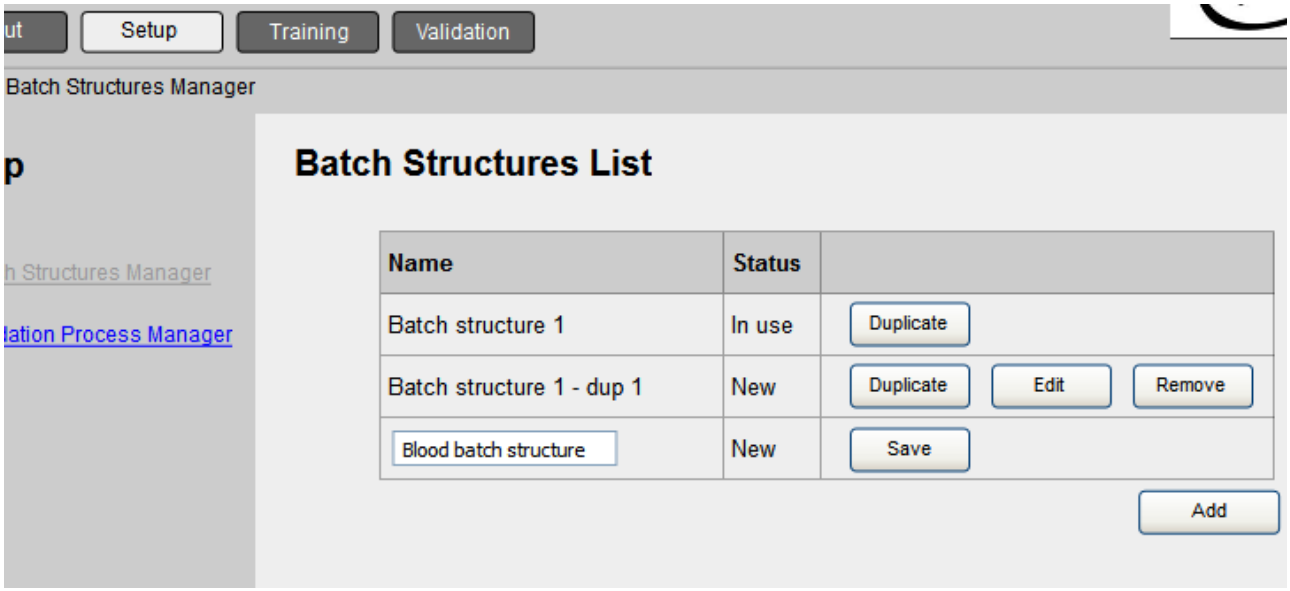


Figura 2: Exemplo de criação de uma estrutura de lote

Após a criação de uma nova estrutura de lote vazia, o usuário deve editá-la para adicionar as estruturas de arquivo que compõem a estrutura de lote em questão, como mostrado na figura 3. A criação de uma estrutura de arquivo ocorre da mesma forma que a criação de uma estrutura de lote, o usuário fornece o nome do arquivo e uma estrutura de arquivo vazia é criada.

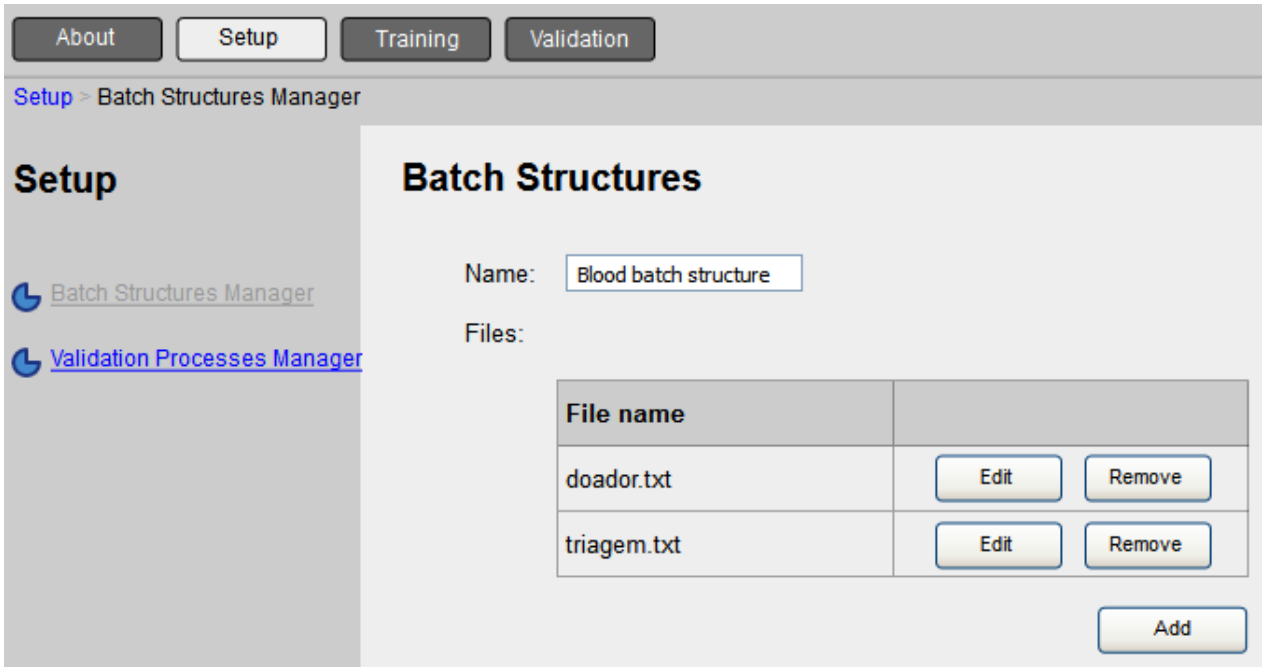


Figura 3: Exemplo de edição de uma estrutura de lote

Uma vez que uma estrutura de arquivo vazia tenha sido criada, o usuário deve editá-la para adicionar o nome e tipo de cada um de seus atributos, como ilustrado na figura 4. Também é permitida a troca da ordem dos atributos entre si, além da edição e remoção de atributos. O usuário também deve informar como o arquivo CSV deve ser lido dentro do lote, quais são os separadores dos campos, codificação do arquivo, delimitador dos campos e se há um cabeçalho na primeira linha ou não.

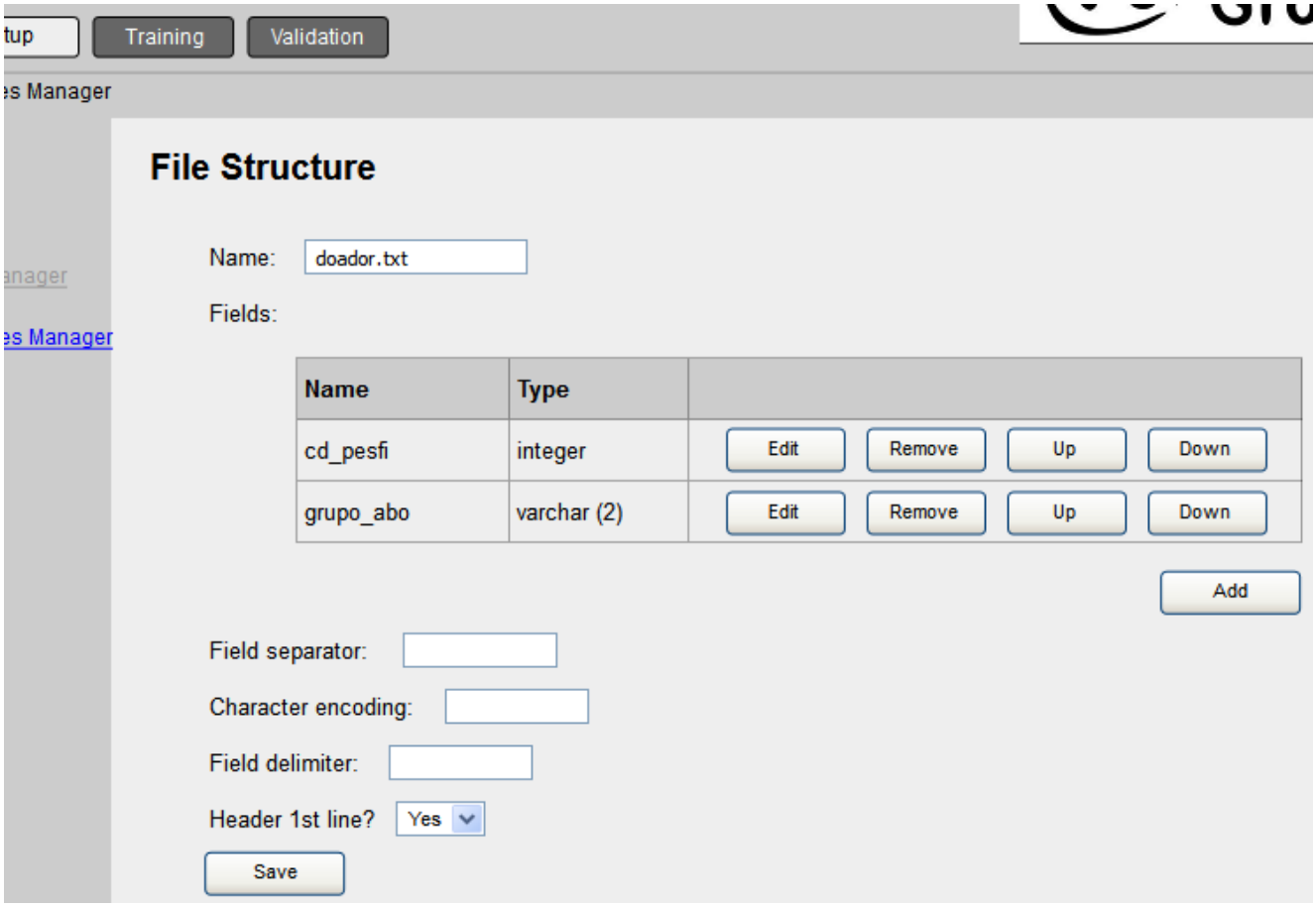


Figura 4: Exemplo de edição de uma estrutura de arquivo

4.1.2 Validador

Os validadores lógicos de um processo de validação são criados via interface na tela de edição de processos de validação, como mostrado na figura 5.

Além disso, um validador lógico tem também um estado, que pode ter o valor "novo", ou seja, sem dados de treinamento atrelados a seus validadores físicos ou, caso contrário, o estado tem o valor"em uso". Quando um usuário cria um validador lógico, uma nova instância é adicionada na lista de validadores lógicos do processo de validação indicando um validador lógico novo e inicialmente vazio. Para configurar o validador lógico, o usuário deve editá-lo. A edição de um validador lógico só pode ocorrer enquanto o estado deste é novo, caso contrário, o validador está em uso e sua edição causa a quebra de consistência entre os lotes já classificados antes da edição e os classificados após, portanto a edição de validadores em uso não é permitida.

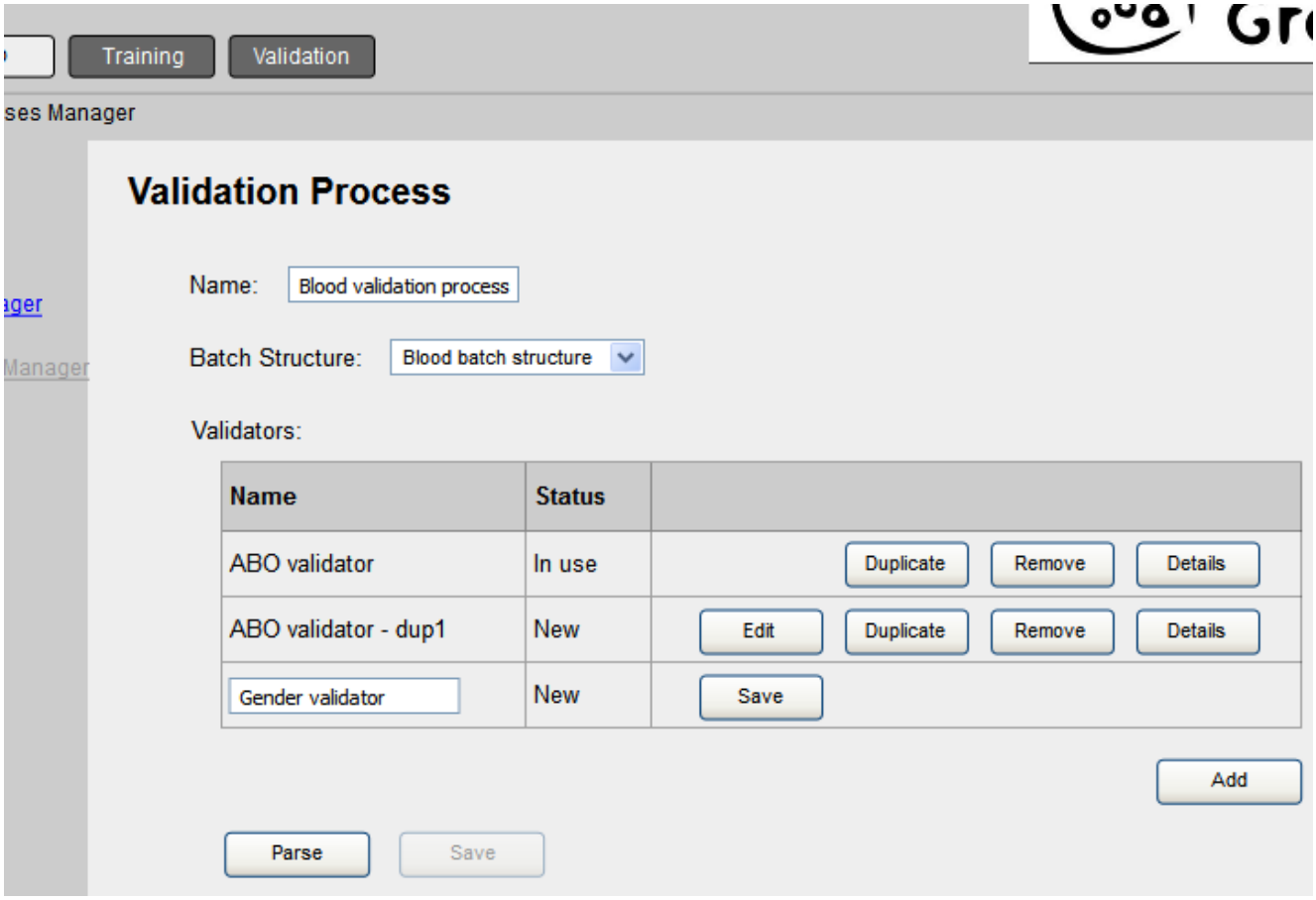


Figura 5: Exemplo de criação de um validador lógico

A figura 6 mostra a edição de um validador lógico, o usuário deve escolher qual o atributo comportamental do validador. Assim, o usuário seleciona um arquivo da estrutura de lote usada pelo processo de validação que contém o validador lógico e serão mostrados na interface todos os atributos do arquivo selecionado. Então o usuário seleciona qual o atributo comportamental e configura suas categorias ou intervalos de valores possíveis.

Dependendo do problema, pode ser desejável utilizar um tipo específico de algoritmo para detectar anomalias. A interface permite configurar o algoritmo usado pelo validador. Como um determinado contexto pode exigir um algoritmo diferente, no validador lógico é definido apenas o padrão, ou seja, se não for especificado um algoritmo específico para um contexto, a detecção de tal contexto deve usar o algoritmo padrão definido no validador lógico.

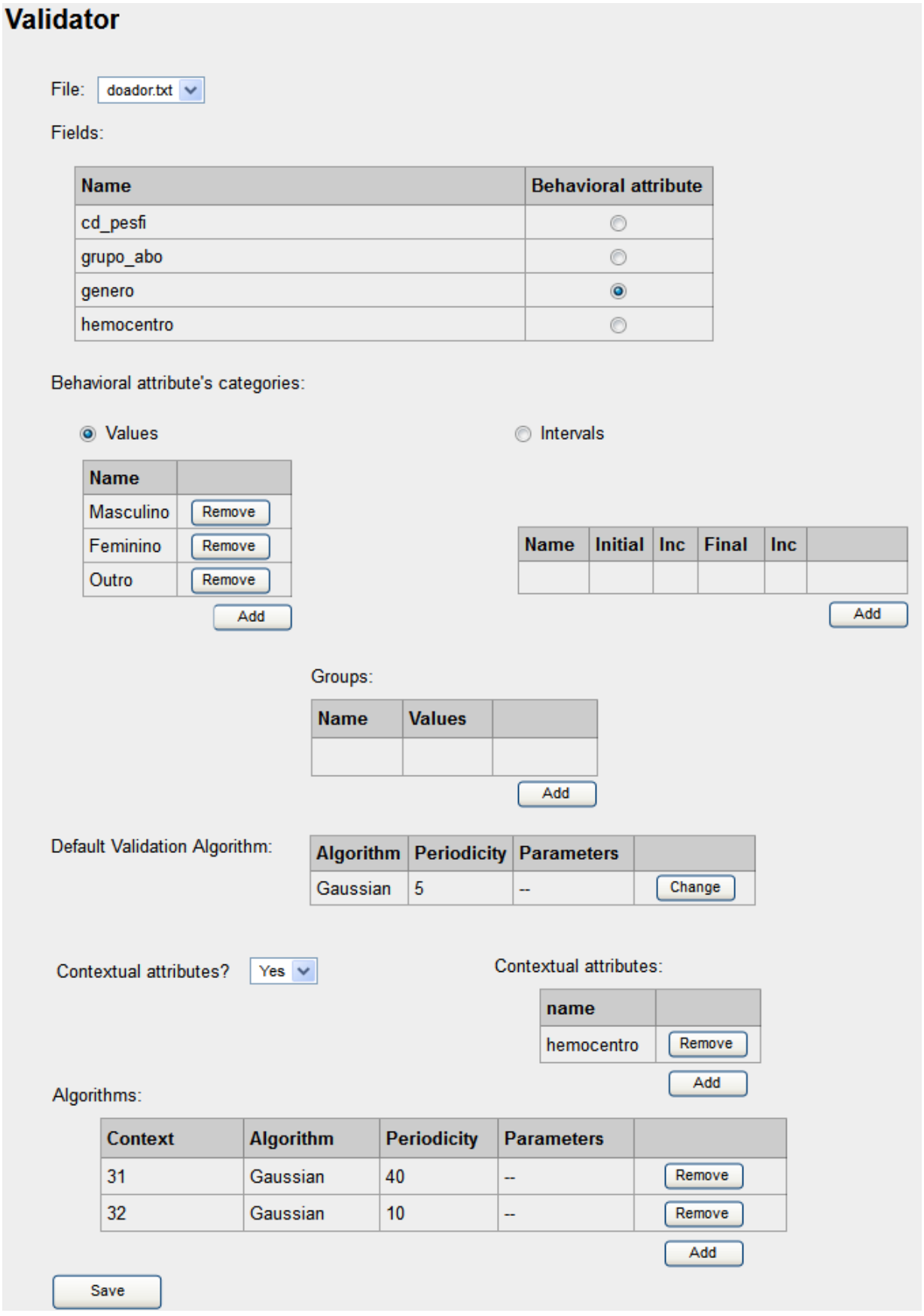


Figura 6: Exemplo de configuração de um validador lógico com atributo contextual

No exemplo da figura 6, o usuário optou por usar um atributo contextual. Quando um

atributo contextual é adicionado, a interface apresenta ao usuário a tela ilustrada na figura 7, na qual permite ao usuário especificar os contextos levados em conta para criar os validadores físicos que compõem o validador lógico da figura 6.

Durante a configuração dos validadores, a interface também permite agrupar categorias do atributo comportamental, assim ao invés de usar as porcentagens de ocorrência das categorias individualmente, o detector usa a soma das porcentagens agrupadas como se fosse uma só categoria. Também é possível agrupar as categorias dos atributos contextuais, como mostrado da figura 7.

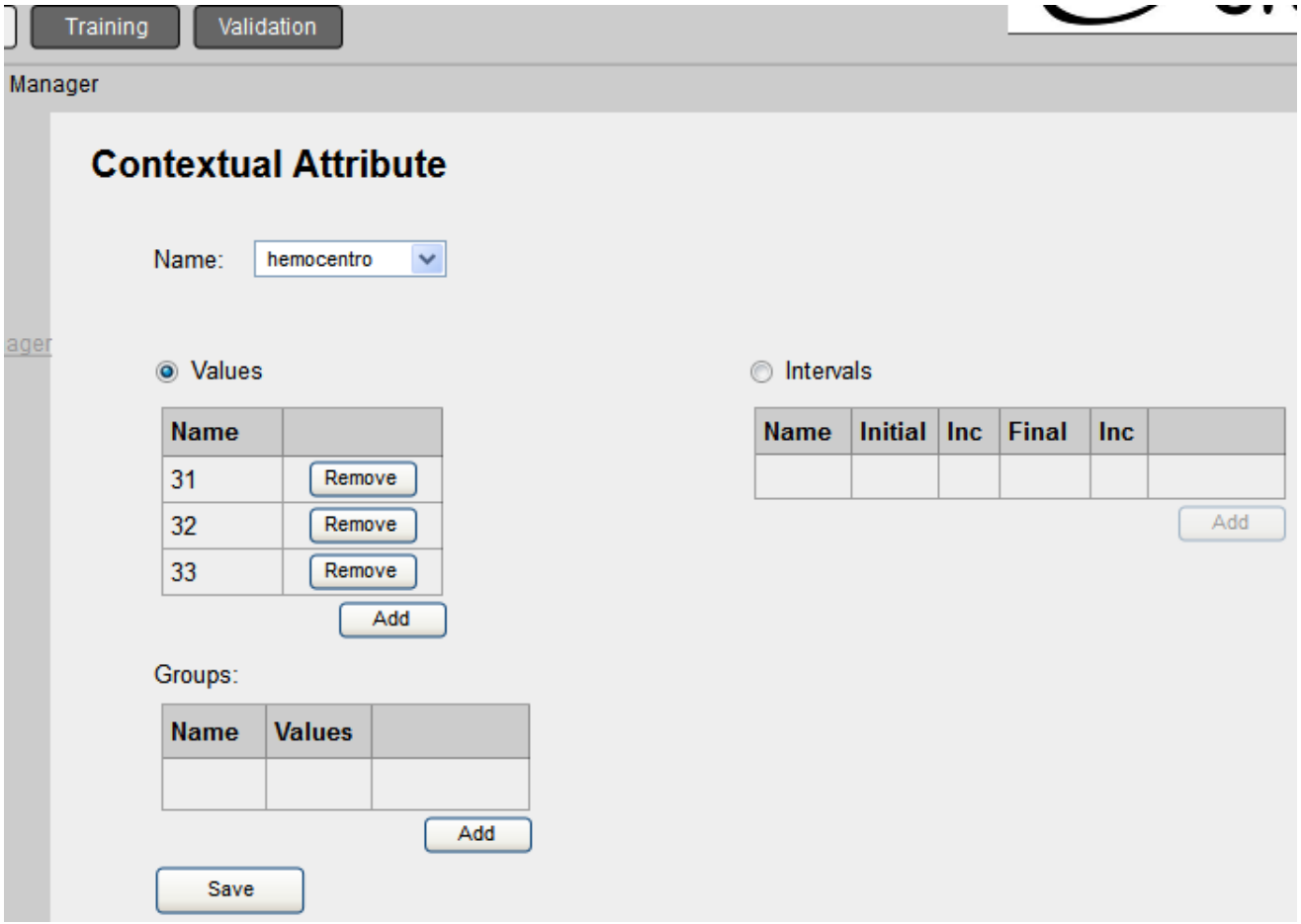


Figura 7: Exemplo de configuração dos contextos definidos por um atributo contextual

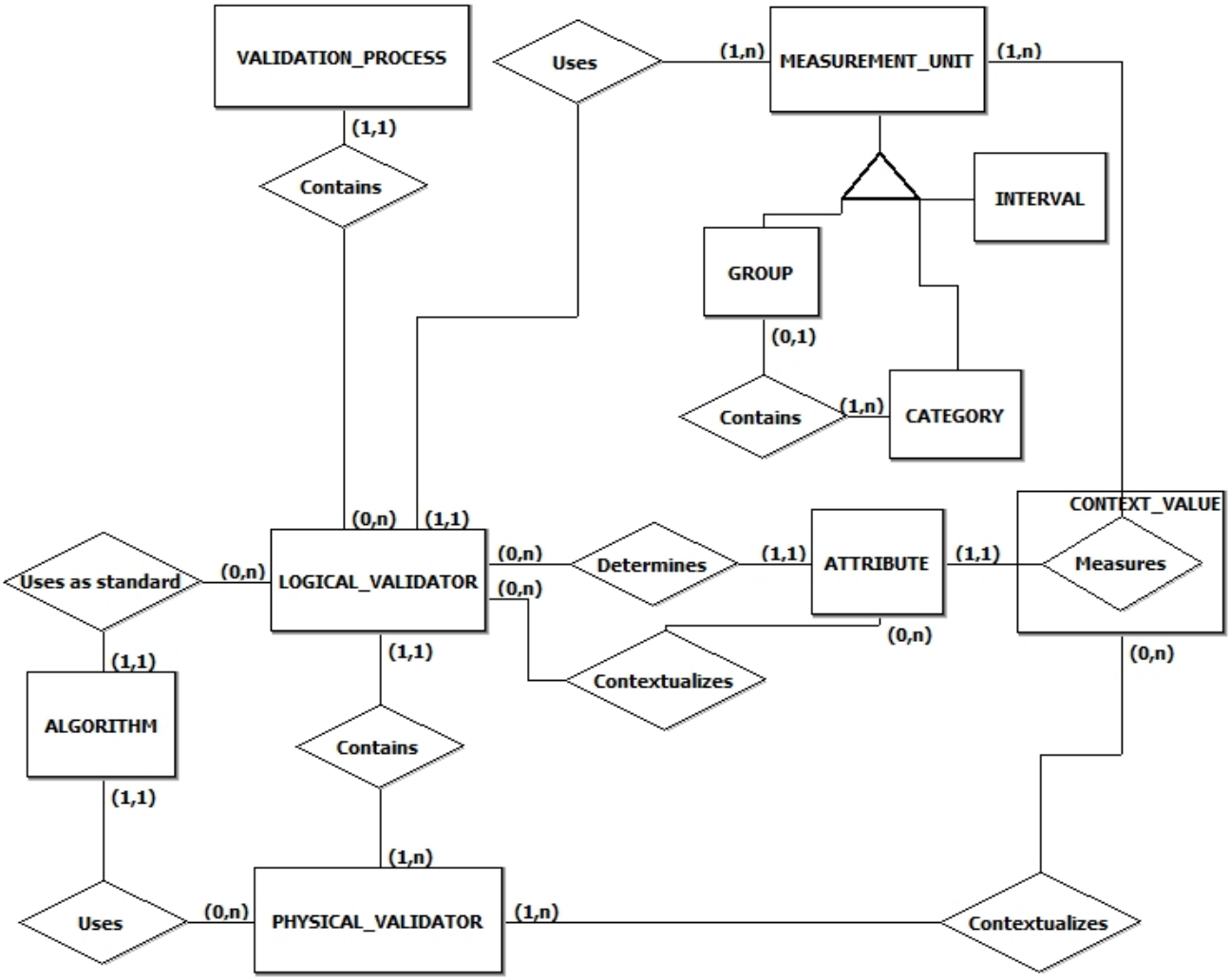


Figura 8: Diagrama da modelagem conceitual do banco de dados para validadores

Na figura 8 a entidade que representa processos de validação (“VALIDATION_PROCESS”) relaciona-se com a entidade que representa validadores lógicos (“LOGICAL_VALIDATOR”) já que um processo de validação contém validadores lógicos. Da mesma forma, a entidade de validadores lógicos se relaciona com a entidade dos validadores físicos (“PHYSICAL_VALIDATOR”), pois um validador lógico contém um ou mais validadores físicos. Estes relacionamentos são realizados usando chave estrangeira, assim cada validador físico contém a chave do validador lógico que o contém e cada validador lógico contém a chave do processo de validação que o contém.

Há também uma entidade para representar algoritmos detectores de anomalias (“ALGORITHM”), que se relaciona com validadores lógicos que guardam a chave do algoritmo que é usado como padrão e com validadores físicos que contém a chave de um algoritmo caso este tenha sido especificado na configuração, ou, caso não tenha, o validador físico contém a chave do algoritmo padrão do validador lógico que o contém.

A entidade de validador lógico também está relacionada à entidade de atributo (“ATTRIBUTE”) de dois modos. Um dos relacionamentos é o de definição de um validador lógico que contém a chave do atributo comportamental. O outro relacionamento é o de contextualização, ou seja, a ligação entre um validador lógico e seus atributos contextuais, feita através de uma tabela de relacionamentos já que um atributo pode contextualizar mais de

um validador lógico e um validador lógico pode ter mais de um atributo contextual.

Outra entidade importante é a de unidade de medida ("MEASUREMENT_UNIT"), que representa o modo de separar os valores dos atributos comportamentais e contextuais de um validador lógico. Deste modo, uma unidade de medida deve estar relacionada a um validador lógico, que pode conter mais de uma unidade de medida, então a unidade de medida contém a chave estrangeira do validador lógico. Uma unidade de medida é uma generalização que pode ser uma categoria ("CATEGORY"), um intervalo ("INTERVAL") ou um grupo de categorias ("GROUP"). Um grupo está relacionado com as categorias que contém, mas uma categoria pode não pertencer a nenhum grupo, portanto o relacionamento é feito por um campo que guarda a chave do grupo na categoria caso haja grupos, caso contrário guarda um valor "-1" marcando que grupos não são aplicados sobre a categoria.

Além disso, unidades de medida estão também relacionadas aos atributos cujos valores são medidos pelas unidades de medida. Todo atributo medido deve ter pelo menos uma unidade para atribuir seus valores e cada unidade de medida deve ser usada por apenas um atributo. Porém esse relacionamento será usado para definir contextos, então ele é definido como uma entidade que representa um valor de contexto ("CONTEXT_VALUE") que contém ambas as chaves do atributo medido e da unidade de medida usada.

Como um validador físico representa a detecção sobre um determinado contexto, a entidade de validador físico deve estar relacionada ao contexto e um contexto é composto de uma ou mais entidades de valores de contexto. Por exemplo, caso os atributos contextuais de um validador lógico fossem "Sexo" e "Hemocentro", um exemplo de valor de contexto seria a tupla (Sexo: Feminino), exemplos de contexto seriam as tuplas (Sexo:Feminino, Hemocentro:31) e (Sexo:Feminino, Hemocentro:32). Assim, como um validador físico pode estar atrelado a mais de um valor de contexto e um valor de contexto pode fazer parte de mais de um contexto, assim relacionando-se a mais de um validador físico, há uma tabela de relacionamento para guardar a chave da entidade valor de contexto e a chave da entidade validador físico.

4.1.3 Gerenciamento de processos de validação

Um processo de validação tem um estado que mede dois fatores, a ativação e o uso. Quando inativo, um processo não pode ser usado para realizar a validação de novos lotes. Um processo pode ser ativado e inativado manualmente a qualquer momento pela interface. Quando um processo está em uso, isso significa que ele já recebeu lotes para validação ou treinamento, caso contrário o processo é considerado novo. Assim, os possíveis estados de um processo de validação são "em uso/ativo", "em uso/inativo", "novo/ativo"e "novo/inativo". A interface permite as seguintes operações sobre os processos de validação:

- **Remoção:** pela qual o processo de validação é removido do sistema assim como seus validadores.
- **Duplicação:** cria um novo processo de validação com as mesmas configurações do processo duplicado, exceto pelo estado do novo processo que independe do estado do processo original e, por padrão, é sempre "novo/inativo".
- **Ativação/Inativação:** permite ao usuário alterar o estado quanto à ativação de um processo de validação.
- **Edição:** altera as configurações de um processo de validação, ou seja, permite modificar seu nome, trocar sua estrutura de lotes e gerenciar sua lista de validadores lógicos.

Se o processo é considerado novo, então sua edição permite a mudança do nome do processo, a troca da estrutura de lotes por outra, a criação de novos validadores, a remoção dos validadores do processo e também a edição dos validadores.

Caso um processo esteja em uso, a edição de seus validadores não é permitida para evitar falta de consistência entre os lotes já recebidos pelo processo e os que futuramente serão recebidos. Portanto, se o processo está em uso, apenas a mudança de nome do processo, a criação de novos validadores e a remoção de validadores do processo são permitidas. A troca de estrutura de lotes em um processo que está em uso é permitida sob algumas ressalvas.

A troca de estrutura de lotes pode causar a quebra de consistência, mas como a nova estrutura pode ser uma expansão da anterior, pode ser que a consistência seja mantida e portanto a troca de estrutura de lote deve ser permitida, mesmo quando o processo está em uso. Assim, ao fim da edição de um processo de validação, é realizada uma verificação para garantir que os validadores lógicos do processo estejam consistentes com a estrutura de lotes.

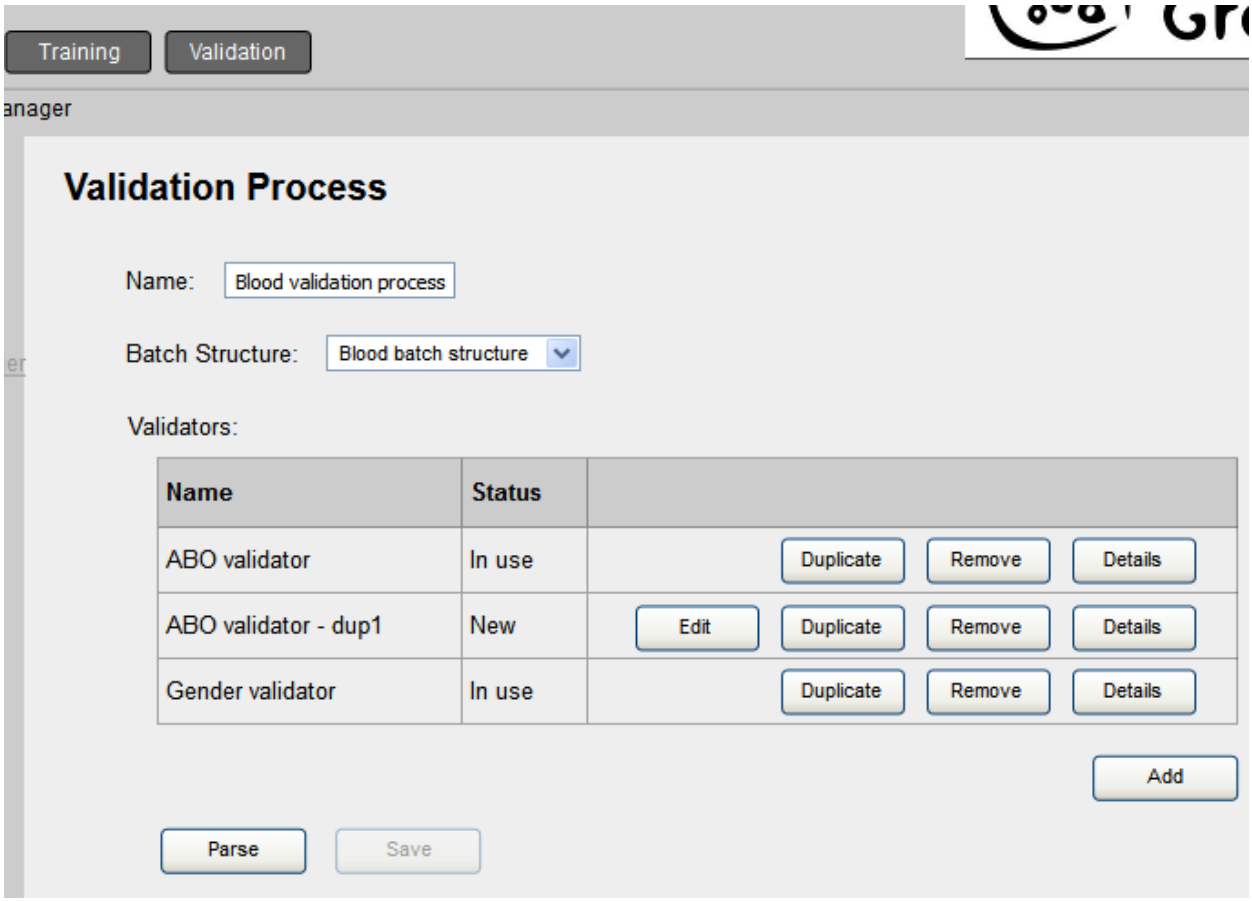


Figura 9: Exemplo de edição de processos de validação estatística.

4.2 Administração do treinamento

O treinamento dos detectores de anomalias pode ser feito de duas maneiras. Pelo envio de lotes classificados por um especialista de domínio, que chamaremos de lotes rotulados, ou através do envio de lotes para validação estatística, que após serem classificados pelo processo de validação são usados no conjunto de treinamento de validações futuras.

Para que sejam usados no treinamento, os lotes reduzidos precisam estar classificados e para armazenar sua classificação possuem um campo "marca". A marca do lote reduzido

pode indicar os seguintes valores:

- **Válido (rotulado):** representa que o lote reduzido foi confirmado como válido por um especialista de domínio.
- **Supostamente válido (não rotulado):** representa que o lote reduzido passou pelo processo de validação e foi considerado válido pelo algoritmo, portanto ele é provavelmente válido, mas não confirmado por especialistas de domínio.
- **Desconhecido (não rotulado):** representa que o lote reduzido foi enviado para treinamento sem passar pelo processo de validação e ainda não recebeu rótulo pelo especialista de domínio.
- **Inválido (rotulado):** representa que o lote reduzido foi confirmado como inválido por um especialista de domínio.
- **Supostamente inválido (não rotulado):** representa que o lote reduzido passou pelo processo de validação e foi considerado inválido pelo algoritmo, portanto ele é provavelmente inválido, mas não confirmado por especialistas de domínio.

Como mostra a figura 10, a interface permite ao usuário alterar a marca de um lote reduzido na tela de gerenciamento de lotes reduzidos. Para realizar a listagem dos lotes reduzidos cujas informações estão registradas no banco de dados o usuário seleciona um validador lógico do processo de validação previamente escolhido e então seleciona o contexto que especifica o lote reduzido, o que também é mostrado na figura 10.

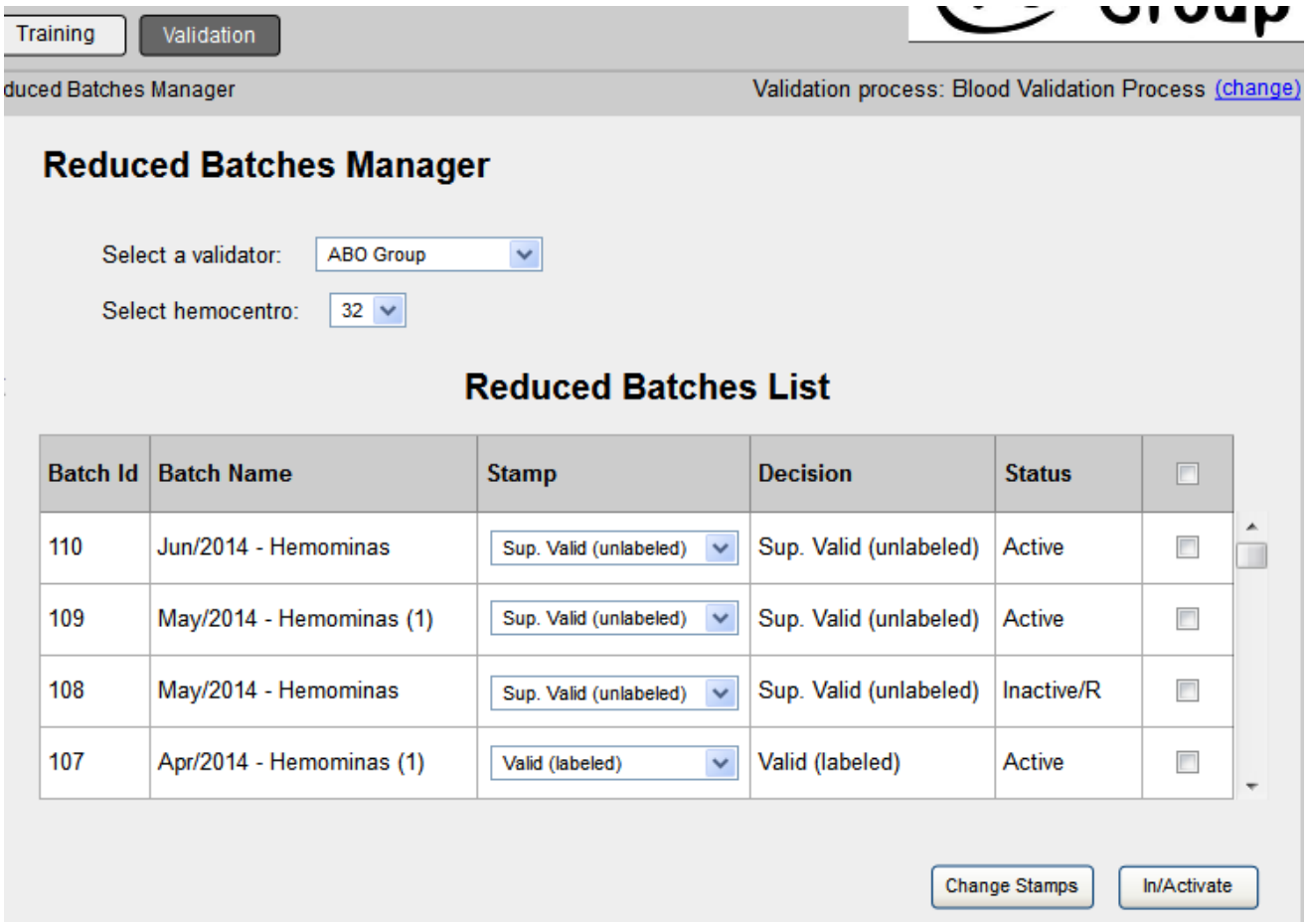


Figura 10: Tela de gerenciamento de lotes reduzidos.

Para o detector de anomalias, lotes reduzidos com as marcas “Válido (rotulado)”, “Supostamente válido (não rotulado)” ou “Desconhecido (não rotulado)” são considerados não anômalos, ou seja, válidos (1). Já os lotes reduzidos com as marcas “Inválido (rotulado)” ou “Supostamente inválido (não rotulado)” são considerados como anomalias, ou seja, inválidos (0).

Os lotes completos podem ser enviados a um processo de validação via interface, como mostra a figura 11. Nela também podemos ver que os lotes possuem marcas, mas estas não podem ser alteradas diretamente via interface, pois são dadas pela conjunção lógica das marcas de seus lotes reduzidos. Assim, caso todas as marcas de seus lotes reduzidos forem consideradas válidas (1), a marca do lote terá valor “válido”, caso uma das marcas de seus lotes reduzidos seja considerada inválida (0), então a marca do lote será “inválido”.

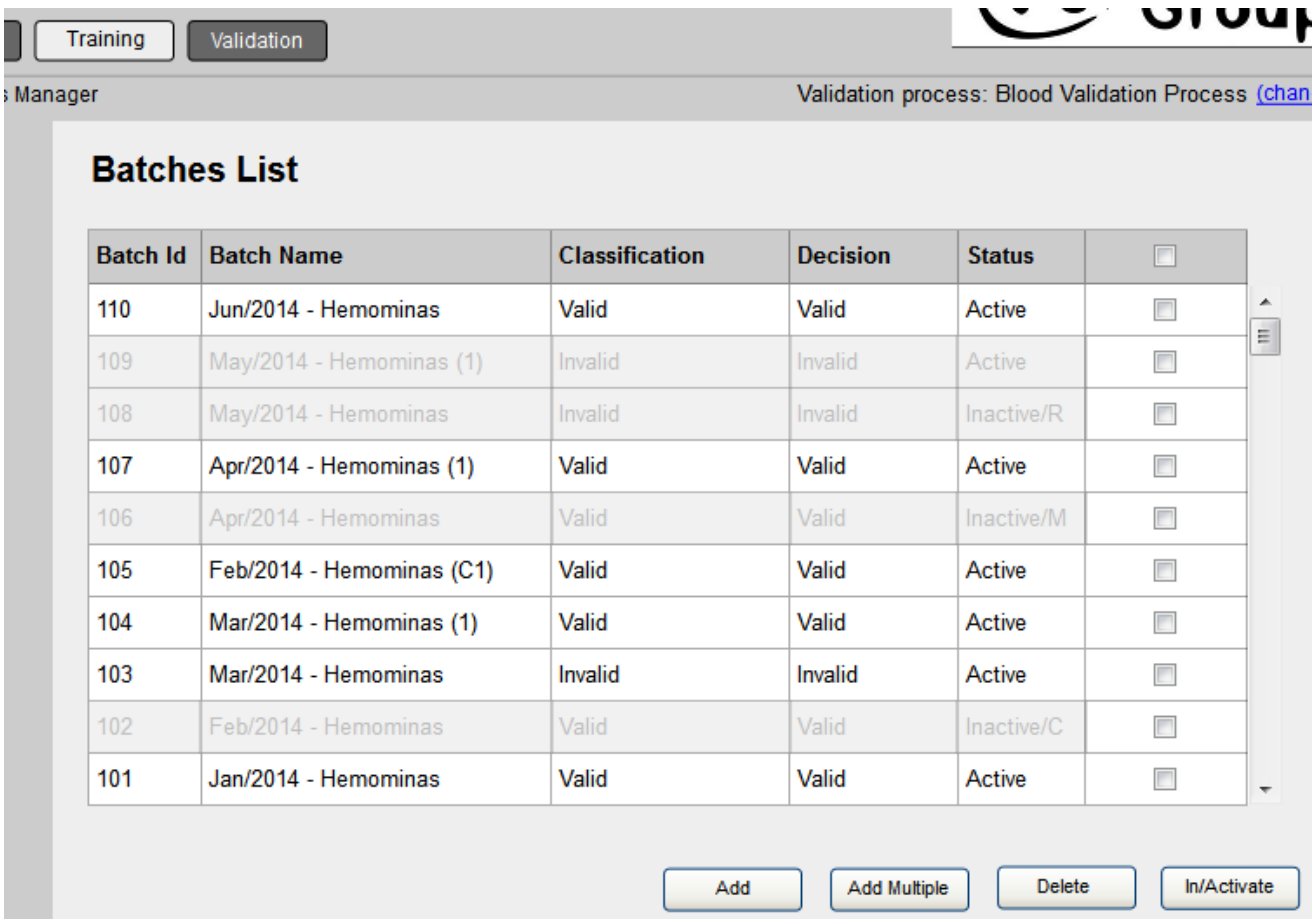


Figura 11: Tela de gerenciamento de lotes.

Como pode ser visto na figura 11, um lote também tem um estado que pode ter valor "ativo" ou "inativo". Um lote inativo deixa de participar do conjunto de treinamento de seus detectores. Os lotes reduzidos também têm seus próprios estados, mas quando um lote está inativo, todos seus lotes reduzidos também estão inativos. Quando um lote está ativo, seus lotes reduzidos podem estar ativos ou inativos. A inativação de lotes ou lotes reduzidos pode acontecer manualmente ou automaticamente.

Para lotes, a inativação automática pode ocorrer por dois motivos. O primeiro caso é quando ocorre a submissão de um lote corretivo, ou seja, um lote que corrige um lote anteriormente enviado, o que pode enviesar o treinamento dos validadores que usam o lote, portanto, sempre que um lote corretivo é enviado para treinamento, o lote corrigido é inativado pelo sistema. O segundo caso também pode ocorrer para lotes reduzidos, é quando ocorre o envio de um lote ou lote reduzido repetido, para evitar o viés causado pela repetição no conjunto de treinamento, todos os lotes ou lotes reduzidos que são repetidos por um lote ou lote reduzido mais atual são inativados pelo sistema.

A interface permite que o usuário mude o estado de lotes ou lotes reduzidos manualmente em alguns casos. A inativação manual de lotes e lotes reduzidos é permitida pela interface, mas caso o lote tenha sido inativado não só manualmente pelo usuário mas também automaticamente pelo sistema e o usuário ativar o lote manualmente, o lote continuará inativo devido à inativação automática.

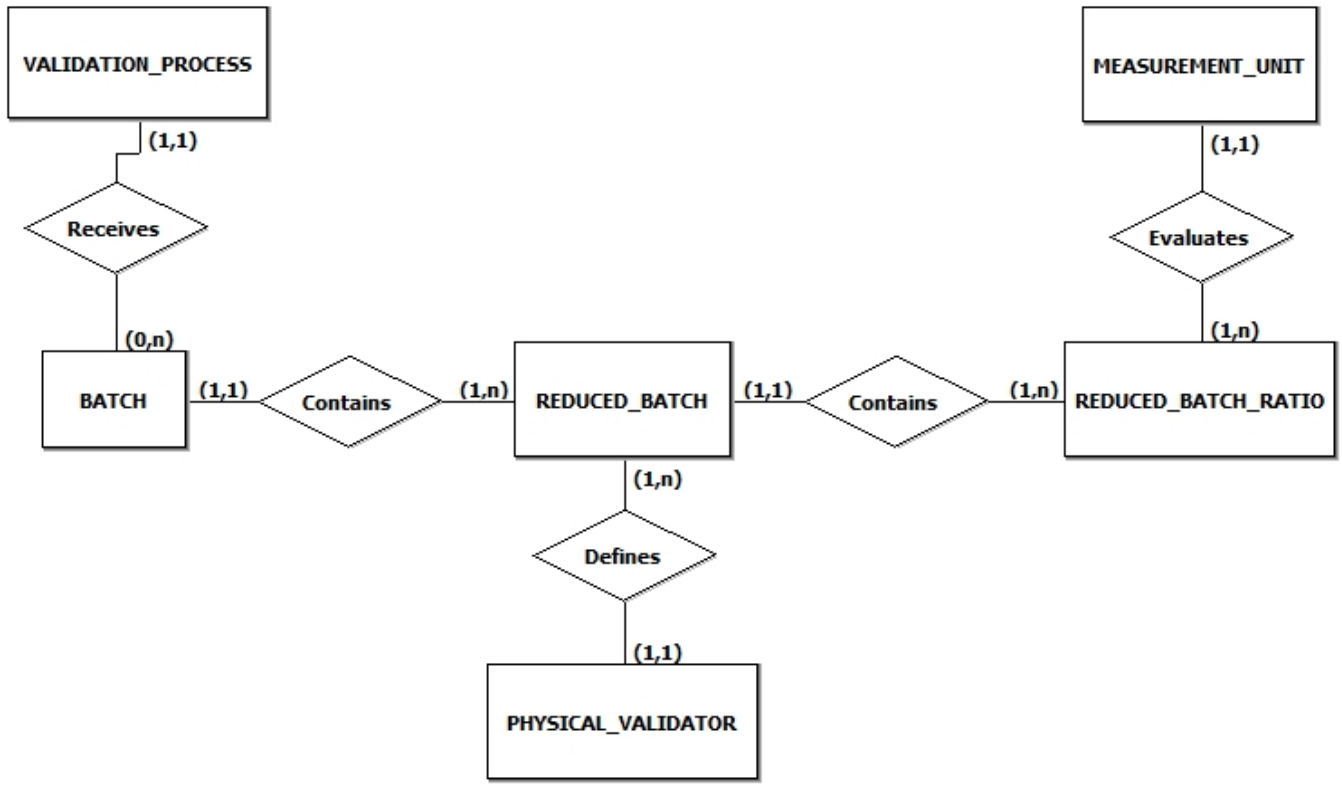


Figura 12: Diagrama da modelagem conceitual das estruturas de gerenciamento dos lotes de treinamento no banco de dados.

No modelo da figura 12, existe uma entidade lote (“BATCH”) que está relacionada à entidade de processo de validação (“VALIDATION_PROCESS”), pois um lote é enviado a um processo de validação específico. Portanto, o lote guarda a chave estrangeira do processo de validação.

Além disso, cada lote contém os lotes reduzidos construídos a partir dos dados do lote original. Assim a entidade de lote reduzido (“REDUCED_BATCH”) está relacionada com a entidade de lote o relacionamento é feito guardando a chave estrangeira da entidade de lote na entidade de lote reduzido.

Como um lote reduzido é usado no treinamento de um detector de anomalias, ele deve ser definido por um validador físico que contém o contexto da detecção e está contido em um validador lógico que contém o atributo comportamental da validação. Portanto, há um relacionamento entre a entidade de lote reduzido e a de validador físico (“PHYSICAL_VALIDATOR”) no qual o lote reduzido mantém a chave estrangeira do validador físico que o define.

É importante para o treinamento dos algoritmos de detecção armazenar as porcentagens de ocorrência de cada unidade de medida em um lote reduzido, sejam categorias, grupos ou intervalos. Deste modo, a entidade porcentagem no lote reduzido (“REDUCED_BATCH_RATIO”) deve estar relacionada não só ao lote reduzido em questão quanto à unidade de medida (“MEASUREMENT_UNIT”), mantendo assim as chaves estrangeiras do lote reduzido e da unidade de medida e também o valor da porcentagem de ocorrência da unidade de medida no lote reduzido.

4.3 Envio de lotes para a validação estatística

Outra funcionalidade oferecida pela interface é o envio de lotes para a validação estatística. O usuário seleciona o processo de validação a ser aplicado sobre o lote e o arquivo do lote a ser enviado. Além disso, o usuário deve indicar se o lote está sendo enviado para corrigir um lote existente com erro e caso esteja, qual o lote que está sendo corrigido. E também deve informar se deseja que o lote seja usado para treinar os algoritmos após ser validado.

Quando o sistema realiza a validação estatística de um lote e o classifica como anomalia ele informa ao usuário que enviou o lote que o lote é supostamente inválido, mas há a possibilidade de algo improvável ter acontecido no mundo real, ou seja, os dados do lote podem representar a realidade. Neste caso, o lote é considerado um lote excepcional e o usuário deve reenviá-lo informando que se trata de um lote excepcional, o sistema então classifica o lote como válido, mas o impede de ser usado no treinamento como lote válido já que seus dados se desviam acentuadamente do padrão e caso fossem usados no treinamento de um algoritmo de detecção distorceriam a distribuição dos lotes de treinamento.

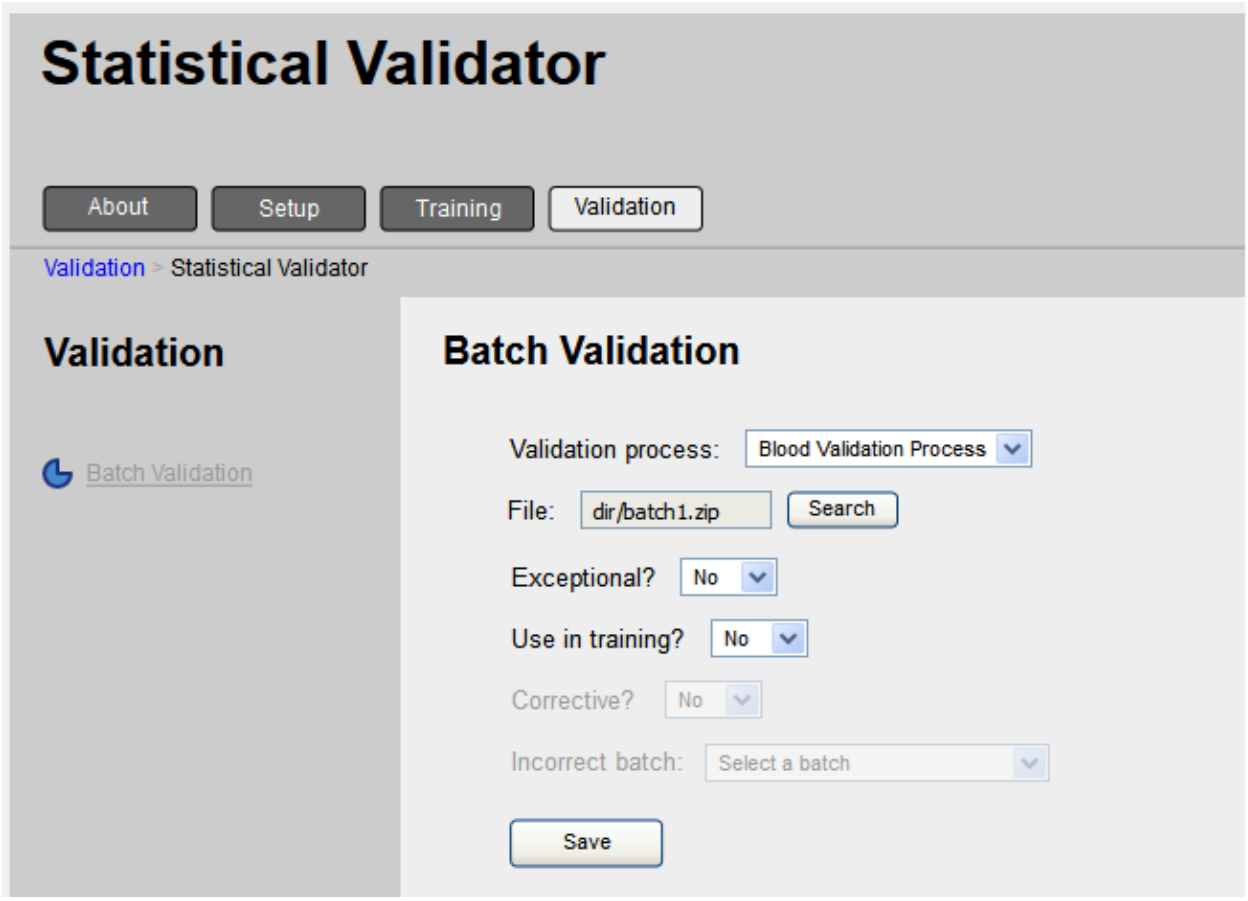


Figura 13: Tela de envio de lotes para a validação estatística.

Após o envio do arquivo, os objetos detectores de anomalias associados a todos validadores físicos que fazem parte do processo de validação selecionado realizam a classificação dos lotes reduzidos criados a partir do lote enviado pelo usuário. Então a interface mostra como resultado a classificação final do lote, que é obtida pela conjunção de todas as classificações provenientes dos objetos detectores usados. Assim, se um dos detectores classificar um lote reduzido como supostamente inválido, o lote é considerado supostamente inválido, se todos

os detectores classificaram seus lotes reduzidos como supostamente válidos, então o lote é considerado supostamente válido. Além disso, o lote e seus lotes reduzidos são armazenados no banco e nos objetos detectores para que possam ser usados como lotes de treinamento em validações futuras.

4.4 Núcleo

O método descrito em [Tak09] sugere um padrão para a implementação de algoritmos detectores de anomalias que fazem parte do núcleo. Tal padrão contém métodos que todos os módulos de detectores devem oferecer. Nesta subseção será apresentada a implementação de tais métodos do algoritmo de detecção gaussiana, apresentado na seção 2.

O padrão consiste na definição dos seguintes métodos:

- módulo.criaDetector: É uma função do módulo que visa criar um objeto da classe do detector. A função recebe como parâmetros o inteiro "número_dimensões" (que representa quantas categorias, grupos ou intervalos estão sendo levados em conta), o inteiro "valor_treinamento" e o booleano "absoluto" (que em conjunto especificam a frequência com a qual o treinamento de um validador é feito, se "absoluto" for verdadeiro então "valor_treinamento" é o número de lotes reduzidos que devem ser adicionados ao detector antes de seu treinamento ser atualizado, caso contrário, "valor_treinamento" é usado como porcentagem do valor total de lotes reduzidos) e um mapa de valores "parâmetros_adicionais" (que pode ser usado para configurar peculiaridades de um algoritmo específico). A função retorna o objeto detector criado com tais configurações.

- detector.adiciona: Este método da classe detector consiste em armazenar uma instância de treinamento, ou seja, um lote reduzido, e sua respectiva marca em uma lista de espera para que esta faça parte do treinamento na próxima vez que o treinamento for realizado. Caso a instância a ser adicionada já esteja no treinamento, a nova instância não é adicionado ao treinamento, mas a marca da instância existente é substituída pela nova marca recebida. O método recebe um objeto "instância" (que contém uma identificação do lote reduzido e um vetor das porcentagens de ocorrência de suas categorias, intervalos ou grupos) e um objeto "marca" (que contém a marca da instância a ser incluída).

- detector.remove: Este método da classe detector é usado para remover uma instância do treinamento e sua respectiva marca. O método coloca a remoção em uma lista de espera e na próxima vez que o treinamento é realizado, a instância em questão é removida. O método recebe uma cadeia de caracteres "id_instância" (que identifica unicamente uma instância no treinamento a ser removida).

- detector.classifica: Este método é usado para realizar a classificação de um lote reduzido utilizando o algoritmo de detecção de anomalias. O método recebe um objeto "instância" (que contém a identificação do lote reduzido a ser classificado e um vetor com suas porcentagens de ocorrência de suas categorias, intervalos ou grupos). O método retorna um booleano "1" caso o lote reduzido seja classificado como "supostamente válido" ou "0" caso o lote reduzido seja classificado como "supostamente inválido".

- detector.estaAtivo: Este método responde se o detector está apto para realizar classificações de instâncias ou não. O método devolve um booleano "1" caso esteja pronto ou "0" caso contrário.

Para implementar um módulo que segue tal padrão, foram mantidas uma lista das instâncias usadas no treinamento e uma fila das operações de adição e remoção de instâncias a serem feitas no próximo treinamento. Além disso, outros métodos de uso interno foram implementados e serão discutidos nos próximos parágrafos.

Como podemos ver no algoritmo 1, a função criaDetector pode ser facilmente implementada chamando o construtor da classe detector passando os parâmetros recebidos e retornando o objeto instanciado.

Algorithm 1 módulo.criaDetector

```
1: function criaDetector(numDimensoes, valorTreinamento, absoluto, parametros)
2:     return Detector(numDimensoes, valorTreinamento, absoluto, parametros)
3: end function
```

Os algoritmos 2 e 3 são similares, ambos incluem um objeto de operação na fila de operações. Operações são objetos compostos por um tipo (uma cadeia de caracteres que vale "add" para adição e "rem" para remoção) e no caso de adição armazenam uma instância e sua marca, no caso de remoção armazenam o id da instância. Após adicionarem a operação na fila, ambos os algoritmos chamam um método interno que checa se a fila foi preenchida o suficiente para que um novo treinamento seja realizado.

Algorithm 2 detector.adiciona

```
1: function adiciona(instancia, marca)
2:     insere adição de instância na fila de operações
3:     self.checaFila( )
4: end function
```

Algorithm 3 detector.remove

```
1: function remove(id_instancia)
2:     insere remoção de instância na fila de operações
3:     self.checaFila( )
4: end function
```

Para checar se um novo treinamento deve ser realizado, o algoritmo 4 verifica se "valorTreinamento" é um valor absoluto ou porcentagem, então o compara da maneira adequada com o tamanho da fila de operações. Se a fila for maior ou igual, então todas as suas operações são realizadas, a fila é esvaziada e então é chamado o método que realiza o treinamento.

Algorithm 4 detector.checaFila

```
1: function CHECAFILA( )
2:     if self.absoluto then
3:         if self.filaDeOperacoes.len ≥ self.valorTreinamento then
4:             for all operação in self.filaDeOperacoes do
5:                 realiza operação sobre a lista de instâncias
6:                 remove operação da fila de operações
7:             end for
8:             SELF.TREINAMENTO( )
9:         end if
10:    else
11:        absValorTreinamento ← self.valorTreinamento * self.listaDeInstancias.len
12:        if self.filaDeOperacoes.len ≥ absValorTreinamento then
13:            for all operação in self.filaDeOperacoes do
14:                realiza operação sobre a lista de instâncias
15:                remove operação da fila de operações
16:            end for
17:            SELF.TREINAMENTO( )
18:        end if
19:    end if
20: end function
```

O método interno que realiza o treinamento é mostrado no algoritmo 5. Para que o treinamento possa ser feito, o detector deve estar ativo. Caso esteja, as instâncias da lista são divididas em dois conjuntos, um de treinamento e um de validação de forma a respeitar as condições de tais conjuntos explicadas na seção 2. Na fase de treinamento, para cada uma das dimensões das instâncias, são calculadas sua média μ e seu desvio padrão σ dentro do conjunto de treinamento. Então, para cada instância no conjunto de validação, é calculado o valor da função de densidade de probabilidade de cada dimensão da instância usando os valores μ e σ para cada dimensão calculados anteriormente, como o algoritmo assume independência entre as dimensões, então a função de densidade de probabilidade conjunta $P(i)$ é calculada a partir do produto das funções de densidade de probabilidade de cada dimensão. Para estimar o valor de ε, o algoritmo pega o máximo e o mínimo dos valores de $P(i)$ obtidos e itera mil vezes no intervalo entre max e min, calculando o F1-Score do algoritmo para cada um destes valores de ε, o valor com o melhor F1-Score é armazenado em self.parametros[“epsilon”]. Fora do pseudocódigo, os cálculos das médias, desvios padrões e funções de densidade de probabilidade são feitos pelas funções oferecidas pelo scipy usando as estruturas de matrizes oferecidas pelo numpy, como visto na subseção 3.2.1.

Algorithm 5 detector.treinamento

```
1: function TREINAMENTO( )
2:     if SELF.ESTAATIVO( ) then
3:         separa a listaDeInstancias em conjuntos de treinamento e de validação
4:         for all dimensão do
5:             calcula a média μ da dimensão no conjunto de treinamento
6:             calcula o desvio padrão σ da dimensão no conjunto de treinamento
7:         end for
8:         for instância i no conjunto de validação do
9:             P(i) ← 1
10:            for all dimensão do
11:                F ← função de densidade de probabilidade da dimensão em i
12:                P(i) ← P(i) * F
13:            end for
14:        end for
15:        min ← minP(i)
16:        max ← maxP(i)
17:        jump ← (max − min)/1000
18:        best ← 0
19:        self.parametros["epsilon"] ← 0
20:        for i in range(1000) do
21:            epsilon ← min + i * jump
22:            calcula F1 usando ε
23:            if F1 > best then
24:                best ← F1
25:                self.parametros["epsilon"] ← ε
26:            end if
27:        end for
28:    end if
29: end function
```

Para verificar se o detector está ativo, o algoritmo 6 verifica as condições necessárias para garantir que o algoritmo possa ser executado e que as classificações feitas pelo algoritmo sejam precisas. Assim, o método checa se o número de instâncias na lista é maior ou igual a um mínimo pré-definido, se existe ao menos uma instância inválida na lista de instâncias para que a etapa de validação funcione e se pelo menos 60% das instâncias da lista são válidas para que façam parte do conjunto de treinamento.

Algorithm 6 detector.estaAtivo

```
1: function estaAtivo( )
2:     minimo ← 100
3:     validas ← quantidade de instâncias válidas da lista de instâncias
4:     invalidas ← quantidade de instâncias válidas da lista de instâncias
5:     if self.listaDeInstancias.len ≥ minimo then
6:         if invalidas ≥ 1 then
7:             if validas ≥ self.listaDeInstancias.len ∗ 0.6 then
8:                 return True
9:             end if
10:        end if
11:    end if
12:    return False
13: end function
```

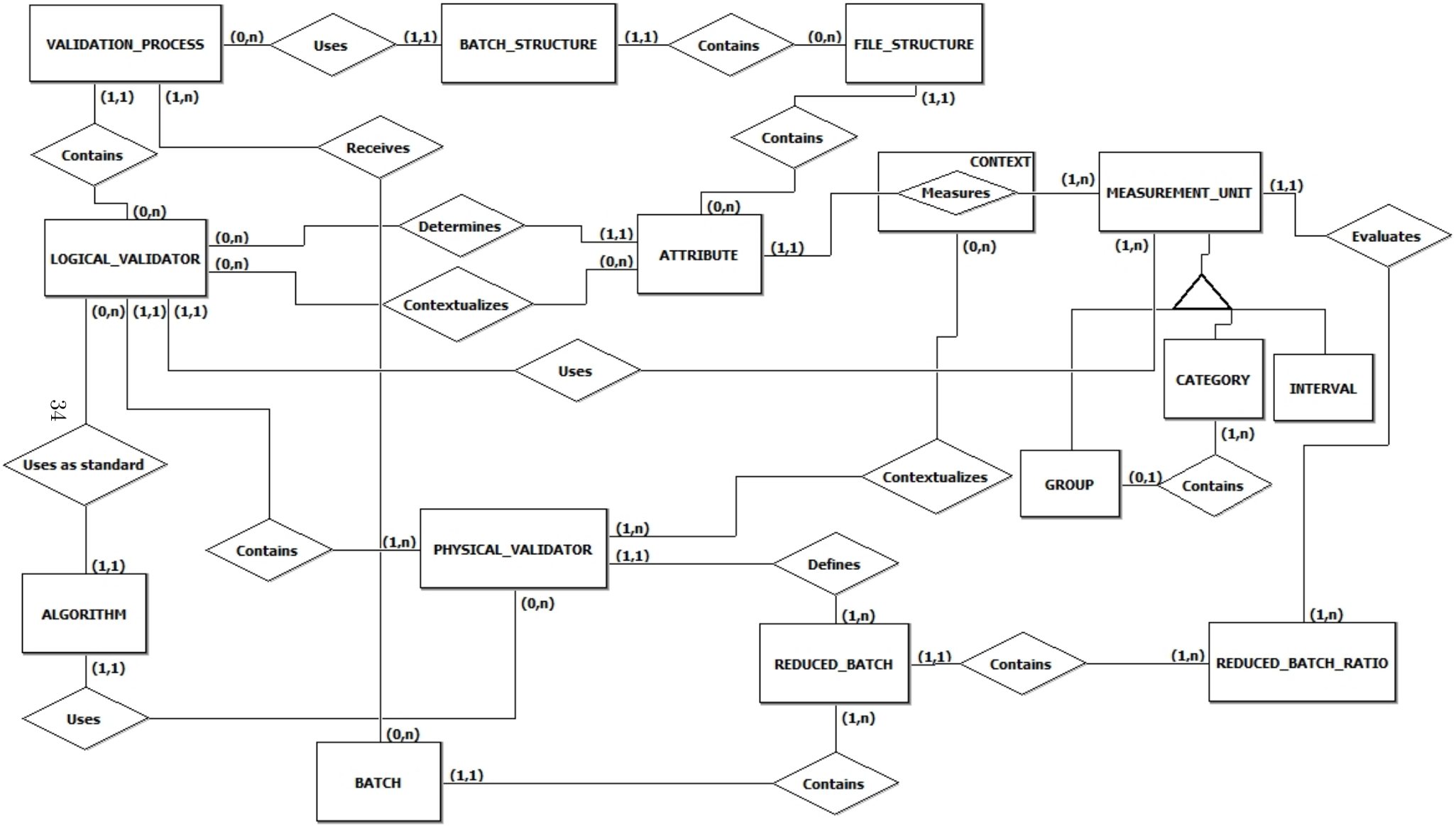
Como mostrado no algoritmo 7, o método "classifica" recebe uma instância e para cada dimensão calcula sua função de densidade de probabilidade usando μ e σ de cada dimensão e como a detecção gaussiana assume independência entre dimensões, então a função de densidade de probabilidade conjunta P é calculada a partir do produto dos valores das funções de densidade de probabilidade de cada dimensão. P é então comparada com self.parametros["epsilon"], que é o melhor valor de ε encontrado após a validação. Caso P seja maior ou igual a self.parametros["epsilon"], o algoritmo retorna verdadeiro, o que representa que a instância é classificada como válida, caso contrário retorna falso, o que representa que a instância é classificada como inválida.

Algorithm 7 detector.classifica

```
1: function classifica(instância)
2:     P ← 1
3:     for dimensão do
4:         Usa μ e σ da dimensão para calcular a função de densidade de probabilidade F
       da instância
5:         P ← P ∗ F
6:     end for
7:     if P >= self.parametros["epsilon"] then
8:         return True
9:     end if
10:    return False
11: end function
```

4.5 Modelagem conceitual

A figura a seguir contém a modelagem conceitual completa do banco de dados usado no sistema de validação estatística configurável. Para prevenir distorções na imagem, a figura foi rotacionada.


VALIDATION_PROCESS
(0,n)
Uses
(1,1)
BATCH_STRUCTURE
(1,1)
Contains
(0,n)
FILE_STRUCTURE
(1,1)
(1,n)
Contains
Receives
Contains
(1,1)
CONTEXT
Measures
(1,n)
MEASUREMENT_UNIT
(1,1)
Evaluates
(0,n)
(0,n)
LOGICAL_VALIDATOR
(0,n)
(0,n)
Determines
Contextualizes
(1,1)
(0,n)
ATTRIBUTE
(1,1)
(0,n)
(1,n)
(0,n)
(1,1)
(1,1)
Uses
CATEGORY
INTERVAL
(1,n)
Uses as standard
Contextualizes
GROUP
(0,1)
Contains
(1,1)
Contains
(1,n)
PHYSICAL_VALIDATOR
(1,n)
(1,1)
Defines
(0,n)
(1,n)
(1,n)
ALGORITHM
(1,1)
REDUCED_BATCH
(1,1)
Contains
(1,n)
REDUCED_BATCH_RATIO
(1,1)
Uses
(0,n)
(1,n)
BATCH
(1,1)
Contains

5 Conclusão

Devido à complexidade da interface, o trabalho de planejamento do modelo e de como suas estruturas são usadas na implementação levou bastante tempo e portanto o sistema ainda está em desenvolvimento. Apesar disso, foi implementada uma versão funcional que realiza a validação estatística, mas de forma incompleta.

A implementação realizada usa os algoritmos apresentados na subseção 4.4 como um módulo de detecção gaussiana do núcleo. Porém as especificação da estrutura de lote e dos validadores lógicos e físicos ainda não estão implementadas. A versão implementada considera uma configuração fixa (hard-coded). Além disso, o envio de lotes de treinamento ainda precisa ser implementado. Portanto, os lotes usados para treinamento são gerados por um programa python a partir de uma distribuição normal, com algumas anomalias no meio.

Os próximos passos do projeto visam concluir a implementação adicionando a configuração e o gerenciamento do treinamento como planejados na seção 4. Além disso, pesquisar e expandir a gama de algoritmos de detecção de anomalias disponíveis ao usuário para permitir que lotes que seguem outro tipo de distribuição possam ser validados estatisticamente, já que os únicos algoritmos usados no sistema supõem que as porcentagens de ocorrência das categorias nos lotes seguem uma distribuição normal.

Referências

[CBK09] V. Chandola, A. Banerjee, and V. Kumar. Anomaly detection: A survey. *ACM Computing Surveys*, 41(3), July 2009.

[com14] The Scipy community. Scipy 0.14.0 reference, 2014. http://docs.scipy.org/doc/scipy/reference/tutorial/general.html , Visitado em: 21/11/2014.

[dev13] Numpy developers. Numpy, 2013. http://www.numpy.org/ , Visitado em: 21/11/2014.

[Fou14a] Django Software Foundation. Django, 2014. https://www.djangoproject.com/ , Visitado em: 01/11/2014.

[Fou14b] Django Software Foundation. Django faq:general, 2014. https://docs.djangoproject.com/en/1.7/faq/general/#django-appears-to-be-a-mvc-framework-but-you-call-the-controller-the-view-and-the-view-the-template-how-come-you-don-t-use-the-standard-names , Visitado em: 01/11/2014.

[Gro14] The PostgreSQL Global Development Group. Postgresql : About, 2014. http://www.postgresql.org/about/ , Visitado em: 21/11/2014.

[KP88] Glenn E. Krasner and Stephen T. Pope. A cookbook for using the model-view-controller user interface paradigm in smalltalk-80. *Journal of Object Oriented Programming*, 1(3):26–49, August/September 1988.

[Ng14] Andrew Ng. Machine learning, xv. anomaly detection (week 9), 2014. https://class.coursera.org/ml-005/lecture/preview.

[Tak09] P. L. Takecian. *Diretrizes metodológicas e validação estatística de dados para a construção de data warehouses*. PhD thesis, Instituto de Matemática e Estatística Universidade de São Paulo, 2009. Capítulo 5.

Parte II
Parte Subjetiva

6 Dificuldades encontradas

Acredito que a principal dificuldade encontrada ocorreu na fase de planejamento do projeto, foi adaptar um algoritmo de aprendizado computacional para realizar a detecção de anomalias sobre lotes de forma que o usuário pudesse facilmente alterar como a validação é feita, o tipo de lote a ser validado e administrar de algum modo as instâncias de treinamento. Através de muitas reuniões com meus orientadores João Eduardo Ferreira e Pedro Losco Takecian, foram desenvolvidas as estruturas e suas interações que permitissem um modelo conceitual consistente da solução.

Outra dificuldade não necessariamente do projeto, que tive foi a escrita da monografia, achei difícil de decidir o que escrever e como organizar o texto, acabei mudando sua estrutura várias vezes e reescrevendo muitos trechos, tal dificuldade pode talvez ser explicada pela dificuldade que senti em explicar estruturas abstratas como validadores ou pela falta de prática de escrita científica, já que eu não fiz nenhuma iniciação científica. O que me ajudou a superar esse obstáculo foi ler as monografias dos alunos dos anos anteriores no depósito de TCCs disponibilizado na página da disciplina.

7 Disciplinas utilizadas

Nesta seção apresentarei as principais disciplinas que auxiliaram no desenvolvimento do projeto. Além do conteúdo das disciplinas a seguir, também foram úteis ao trabalho algumas habilidades específicas desenvolvidas no decorrer do curso e estagiando na reitoria da USP, tais como familiaridade com a linguagem Python e desenvolvimento usando *web frameworks*.

7.1 MAC0459 - Ciência e Engenharia de Dados

Entre outros importantes tópicos de análise de dados, nesta disciplina foi abordado o tópico de detecção de anomalias, que é um dos principais conceitos deste trabalho.

7.2 MAE0121 - Introdução a Probabilidade e a Estatística I

A disciplina ofereceu toda a base estatística para o estudo dos métodos de detecção de anomalias descritos na parte objetiva como a distribuição gaussiana e o cálculo da matriz de covariância.

7.3 MAC0426 - Sistemas de Bancos de Dados

Os conceitos aprendidos na disciplina de sistemas de bancos de dados foram importantes para criar e utilizar o banco de dados que contém os dados das configurações de validações e dos dados de treinamento. Nesta disciplina obtive familiaridade com a sintaxe SQL usada para comandos e consultas no banco PostgreSQL. Além disso, também usei as técnicas de modelagem conceitual de bancos de dados aprendidas na disciplina.

7.4 MAC0242 - Laboratório de Programação II

Aprendi com essa disciplina grande parte do paradigma de programação orientada a objetos, que foi usado amplamente na programação da interface. Além disso, esta foi uma das poucas disciplinas na qual realizei um projeto que durou o semestre todo, o que me deu uma certa idéia de como é trabalhar em um projeto relativamente longo.

8 Conhecimentos adquiridos

Os principais conhecimentos adquiridos foram sobre técnicas de aprendizado de máquina aplicados à análise de dados seguindo o roteiro de estudo de [Ng14] com foco no capítulo de detecção de anomalias.

Além disso, a implementação da interface se trata de minha primeira experiência com o *web framework* Django e com o sistema gerenciador de banco de dados PostgreSQL.

9 Agradecimentos

A princípio, gostaria de agradecer aos meus orientadores João Eduardo Ferreira e Pedro Losco Takecian pela oportunidade, pelo suporte oferecido, por todo o tempo dedicado a guiar o trabalho na direção certa e por tornarem possível realizar este projeto.

Agradeço também à minha família pelo constante apoio em tudo que eu faço e aos amigos do BCC que fizeram da graduação uma experiência inesquecível.