# Análise dos Experimentos

Este experimento avalia o desempenho da classificação de imagens médicas realizada pelos algoritmos Relevância Binária e o KNN multirrótulo. A seguir é apresentado uma breve descrição dos recursos utilizados, resultados e conclusões.

## Experiments01

### Atributos da classe

* folder: File
* listOfFiles: File[]
* csvName: String
* labels: TreeMap<String, String>
* relationName: String
* irmaCodeStructureName: String
* dataset: MultiLabelInstances
* classifiers: MultiLabelLearnerBase[]
* imageQuantity: int
* clusterQuantity: int
* buildArff: Boolean
* getKeyPoints: Boolean
* keyPointsQuantity: int
* classify: boolean
* SIFTHistogram: TreeMap<String, int[]>
* log: String
* register: LogBuilder

### Métodos da classe

* init(): boolean
* checkExistingFiles(): void
* getImages(): void
* getLabels(): void
* getImagesKeypoints(): ArrayList<String>
* removeAttribute(String, int): String
* getKeyPointsClusters(): int[]
* getSIFTFeatures(): void
* getGaborFeatures(): void
* buildArff(): void
* instantiateClassifiers(): void
* runCrossValidadte(): void

### Resultados

O experimento apresenta problemas durante a execução. No método *getSIFTFeatures*() sempre ocorre problema de estouro de memória na chamada do método *getDataSet* do objeto *ArffLoader*.

### Conclusão

O experimento necessita ser reestruturado para melhor consumo de memória.

Organizar os membros relacionados em classes distintas pode ser uma solução.

## Experiments01.2

### Atributos da classe

#### Conjunto de Imagens

* folder: File
* listOfFiles: File[]
* csvName: String
* labels: TreeMap<String, String>

#### Arquivo Arff

* relationName: String
* irmaCodeStructureName: String
* dataset: MultiLabelInstances
* buildArff: Boolean

#### Classificadores

* classifiers: MultiLabelLearnerBase[]
* classify: boolean

#### Características SIFT

* imageQuantity: int
* clusterQuantity: int
* getKeyPoints: Boolean
* keyPointsQuantity: int
* SIFTHistogram: TreeMap<String, int[]>

#### Log de Eventos

* log: String
* register: LogBuilder

### Métodos da classe

* init(): Boolean

#### Conjunto de Imagens

* checkExistingFiles(): void
* getImages(): void

#### Arquivo Arff

* getLabels(): void
* buildArff(): void

#### Características SIFT

* getImagesKeypoints(): ArrayList<String>
* removeAttribute(String, int): String
* getKeyPointsClusters(): int[]
* getSIFTFeatures(): void

#### Características GABOR

* getGaborFeatures(): void

#### Classificadores

* instantiateClassifiers(): void
* runCrossValidadte(): void

### Novas classes

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  | | --- | | **Conjunto de Imagens** | | - folder: File - listOfFiles: File[] - csvName: String  - labels: TreeMap<String, String> | | + checkExistingFiles(): void  + getImages(): void | | |  | | --- | | **Arquivo Arff** | | - relationName: String  - irmaCodeStructureName: String - dataset: MultiLabelInstances - buildArff: boolean | | + getLabels(): void  + buildArff(): void | | | |  | | --- | | **Classificadores** | | - classifiers: MultiLabelLearnerBase[]  - classify: boolean | | + instatiateClassifiers(): void  + runCrossValidate(): void | |
| |  | | --- | | **Características SIFT** | | - imageQuantity: int - clusterQuantity: int  - getKeyPoints: boolean - keyPointsQuantity: int - SIFTHistogram: TreeMap<String, int[]> | | + getImagesKeypoints(): ArrayList<String>  + removeAttribute(String, int): String  + getKeyPointsClusters(): int[]  + getSIFTFeatures(): void | | | |  | | --- | | **Experiments** | | - log: String  - register: LogBuilder | | + init(): boolean | | |

### Conclusão

A reestruturação do Experiments01 é necessária. A organização do experimento em classes separadas pode melhorar o consumo de memória pela JVM, além de facilitar a manutenção do código. Verificou-se que reestruturar o experimento partindo do código já escrito poderia levar mais tempo do que reescrever o código novamente. Além disso, maiores esforços estariam concentrados em adaptações para que os códigos reestruturados se encaixassem, ao invés dos objetivos que eles deveriam atingir. Reestruturar o Experiments01 em classes separadas, utilizando código original apenas como referência para implementar rotinas similares.

## Experiments01.3

Reestruturação de Experiments01.

### Requisitos técnicos

1. A aplicação deverá ser executada por meio de ambiente texto para que seja possível seu uso no servidor;
2. As entradas deverão ser consumidas de um arquivo para que seja possível a automatização da aplicação;
3. Separar as fases do experimento em classes para que satisfaça os requisitos da reestruturação do **Experiments01**. Para este experimento estão previstas as seguintes fases:
   1. Obtenção do conjunto de amostras;
   2. Extração de características das amostras;
   3. Avaliação dos classificadores.

### A aplicação

Dadas as fases do experimento, descritas no item 3 dos Requisitos técnicos, as seguintes classes deverão ser implementadas:

* Amostras - essa classe abrange todos os procedimentos relacionados com a fase de obtenção do conjunto de amostras;
* Características - essa classe abrange todos os procedimentos relacionados com fase de extração de características das amostras;
* Classificadores - essa classe abrange todos os procedimentos relacionados com a fase de avaliação dos classificadores.

### A classe Amostras

* Método *obtem*() - Fase de obtenção do conjunto de amostras:
  + O sistema deverá receber como entrada um nome que identificará o conjunto de amostras a ser utilizado;
  + Se esse conjunto já existir na pasta do sistema, então não será necessário obter um novo conjunto de amostras;
  + Caso o conjunto não exista, o sistema deverá ter recebido como entrada:
    - O caminho para a pasta que contém as imagens/amostras do conjunto;
    - O caminho para o arquivo que contém a relação nome da imagem/código IRMA;
    - O caminho para o arquivo que contém as estrutura atual de códigos IRMA.
  + Esse método chama o método *obtemHistogramaSIFT*() da classe *Características*. Com o retorno de *obtemHistogramaSIFT*(), é construído o conjunto de amostras.

#### Entradas

* Método *obtem*() - tipo *String* - Caminho para pasta que contém as imagens;
* Método *obtem*() - tipo *String* - Caminho para o arquivo que contém a relação imagem/código IRMA;
* Método *obtem*() - tipo *String* - Caminho para o arquivo com a estrutura atual de códigos IRMA.

#### Saídas

* Tipo *ARFF* - método *obtem*() - Conjunto de amostras utilizado para avaliar os classificadores.

### A classe Características

* Fase de extração de características das amostras;
* Método *obtemPontosChave*() - Obtém os pontos-chave das imagens por meio da técnica SIFT:
  + O sistema deverá receber um limite de pontos-chave que serão obtidos para cada imagem;
  + Cada ponto-chave é representado por um vetor de 128 características;
  + Cada ponto-chave é uma amostra de um novo conjunto auxiliar de dados que é criado para obter as características de histograma SIFT. Nesse conjunto, as imagens são os rótulos das amostras;
  + Os rótulos não devem constar no conjunto de dados auxiliar quando for realizado o agrupamento dos pontos-chave.
    - No entanto, o **Experiments01** mostrou que não há memória suficiente para instanciar o conjunto de dados construído e depois remover o referido atributo.
    - Uma solução seria criar dois conjuntos auxiliares de dados (um com os rótulos e o outro sem) e usar um de cada vez. Isso é importante porque deve ser preservada a ordem das amostras, pois o algoritmo utilizado fornece o identificador do grupo para a posição da amostra. Desse modo, não daria para ser utilizada a lista de nomes das imagens, pois é conhecido apenas o número máximo de pontos-chave obtidos para cada imagem e não a quantia exata.
  + Assim, a execução desse método resultará na criação de dois arquivos, um conjunto auxiliar de dados (*ARFF*) e uma lista de rótulos (*LABELS*). **\***
* Método *obtemHistogramaSIFT*() - Agrupa os pontos-chave e constrói um histograma:
  + O sistema verifica se existe o conjunto auxiliar de dados (*ARFF*) e a lista de rótulos (*LABELS*). Se esses arquivos não existir, chama o método *obtemPontosChaves*();
  + Os centróides de cada grupo são utilizados como identificação dos *bins* que compõe o histograma SIFT;
  + O sistema deverá receber o tamanho do histograma (número de grupos a serem formados);
  + O algoritmo de agrupamento será executado com as amostras do conjunto auxiliar. O parâmetro *“-fast*” deve ser utilizado;
  + **\*** Após a formação dos grupos, o algoritmo retorna o identificador do grupo que foi atribuído a cada amostra do conjunto auxiliar. Nesse momento, as características das amostras não são mais necessárias, apenas seus respectivos rótulos. Portanto, não é necessária a criação de dois conjuntos auxiliares (*ARFF*), mas sim um conjunto auxiliar sem os rótulos e uma lista de rótulos na mesma ordem em que as amostras foram inseridas nesse conjunto.
  + Com os identificadores dos grupos e a lista de rótulos será criado o conjunto de amostras utilizado para avaliação dos classificadores.

#### Entradas

* Método *obtemPontosChave*() - tipo *int* - Limites de pontos-chave obtidos;
* Método *obtemHistogramaSIFT*() - tipo *int* - Tamanho do histograma;

#### Saídas

* Tipo *ARFF* - método *obtemPontosChave*() - Conjunto auxiliar de dados sem os rótulos das amostras;
* Tipo *LABELS* (*txt*) - método *obtemPontosChave*() - Lista de rótulos na respectiva ordem das amostras do conjunto auxiliar;
* Tipo *int*[] - método *obtemHistogramaSIFT*() - Identificadores do grupo atribuído a cada amostra do conjunto auxiliar;

### Projeto da classe Classificadores

* Método *avalia*() - Fase de avaliação dos classificadores:
  + O sistema chama o método *obtem*() da classe *Amostras*;
  + Em seguida, instancia os classificadores que serão avaliados;
  + Executa a avaliação cruzada;
  + Por fim, fornece um arquivo (*results*) com o resultado da avaliação.

#### Entradas

#### Saídas

* Tipo *RESULTS* (*txt*) - método *avalia*() - Resultado da avaliação dos classificadores.

### Arquivo de entrada (conf.ini)

* Arquivo usado para entrada de dados na aplicação.
* Parâmetros definidos:
  + Caminho para a pasta que contém as imagens
  + Caminho para o arquivo que contém a relação imagem/código IRMA
  + Caminho para o arquivo com a estrutura atual de códigos IRMA
  + Limites de pontos-chave obtidos
  + Tamanho do histograma