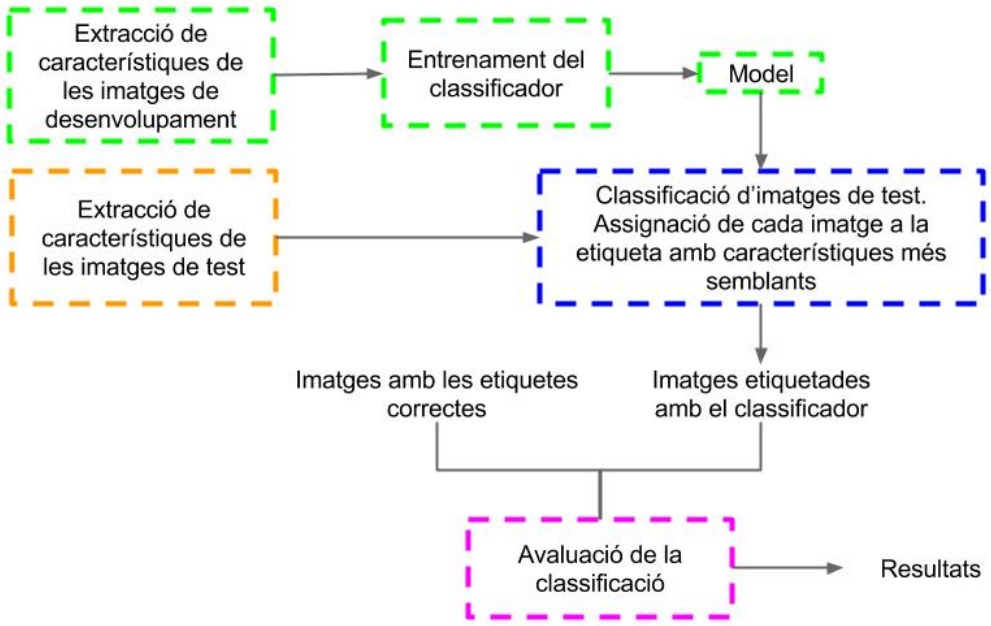
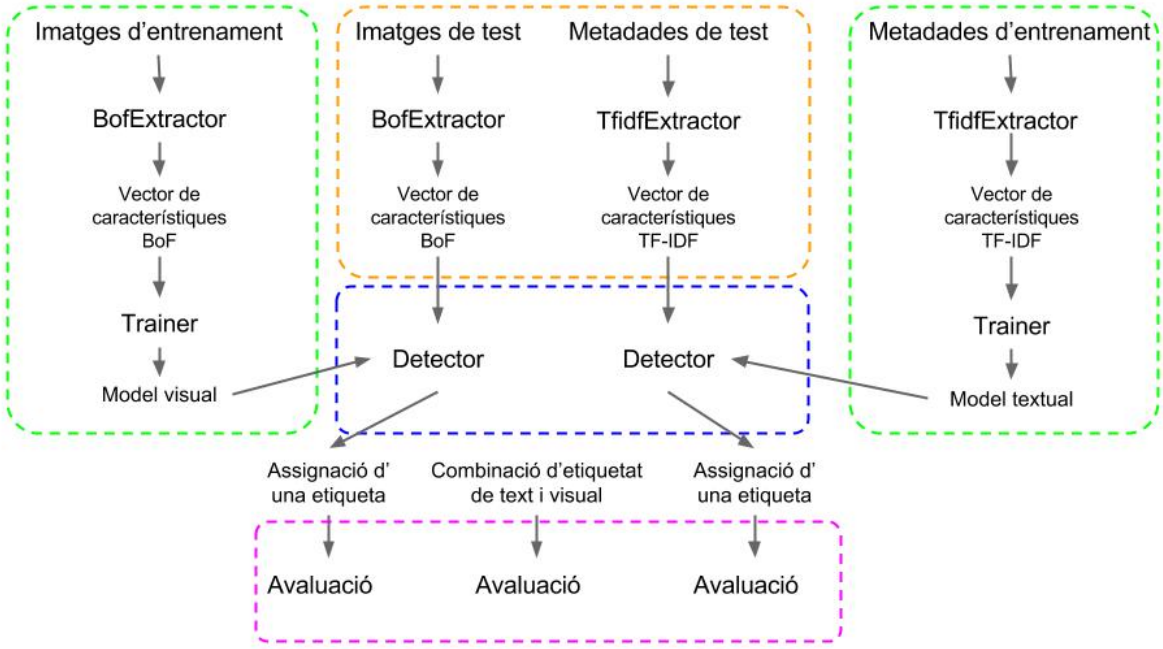
PYXEL

Arquitectura

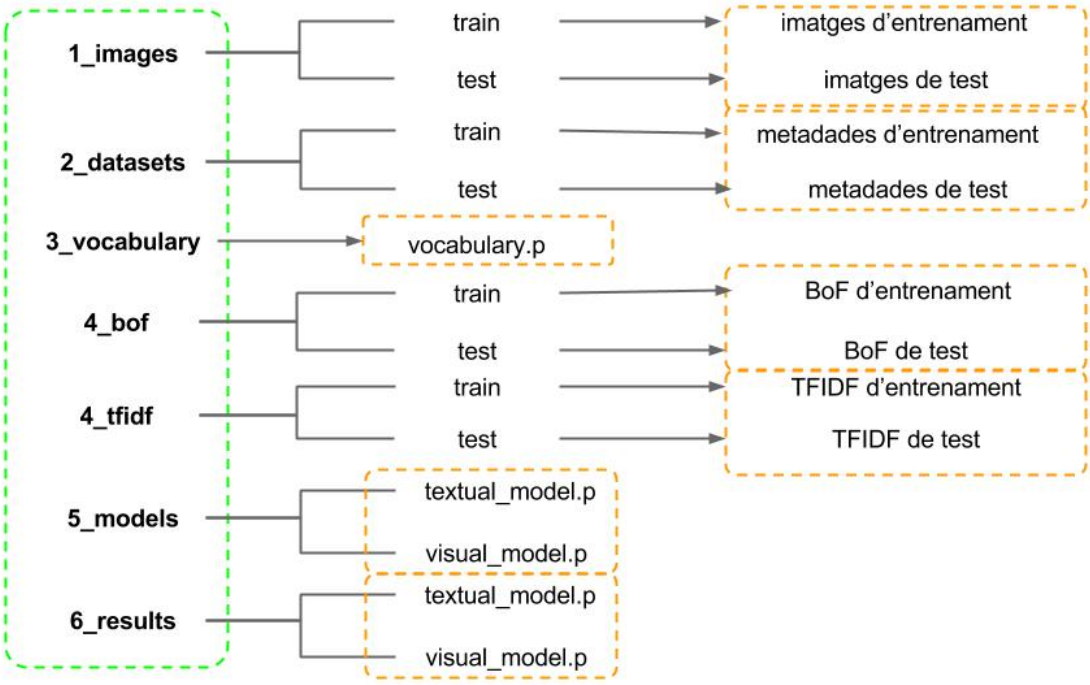
Esquema bàsic del sistema



Estructura del sistema



Esquema dels directoris de treball



Implementació

Per dur a terme la implementació del sistema d'etiquetatge he creat una sèrie de llibreries que faciliten l'accés a altres ja creades amb funcions membre útils per la tasca a desenvolupar. Esta dividida en dos parts, **Modules** i **Tools.**

Modules

La part de mòduls es correspon a la part que anomenaríem llibreries, on es declara cada objecte i les funcions membre que té.
De mòduls tenim els següents:

Computation

Les classes del apartat *Computation* fan referència a la forma de tenir les dades per a facilitar el cost computacional del sistema.

Dataset

Aquesta classe carrega o crea un fitxer de text amb tots els noms root que hi ha en un directori. Va bé per grans volums de dades quan el llistat d'operacions pot ser lent. També pot ser útil per reduir una gran quantitat de memòria en poc espai.

```
Dataset( pathTxtFile, flagSaveInMemory=False, flagVerbose=False)
        pathTxtFile: lloc on es guardarà el fitxer .txt que es genera
```

Funcions

```
def readRootNamesFromTxtFile( pathTxtFile ):
        Funció que serveix per llegir els noms del fitxer
def build( dirFiles, fileExtension ):
        Funció que serveix per construir/crear el fitxer amb els noms de les
        imatges. Com a paràmetres li pasem el directori on es troben les imatges
        dirFiles i l'extensió que tenen fileExtension (.jpg).
def saveToDisk( pathTxtFile, rootNames ):
        Aquesta funció serveis per desar el fitxer.
def getRootNames( ):
        Aquesta funció serveix per obtenir, en memòria els noms de les imatges
        que hi ha al fitxer, o al directori.
```

Classification

Instance

La classe *Instance* conté la informació del nom de la imatge i de la puntuació que li correspon.

```
Intance(docId, score)
        docId: nom de la imatge.
        score: puntuació obtinguda o corresponent.
```

Funcions

```
def getDocId( ):
        Retorna el nom de la imatge.
def getScore( ):
        Retorna la puntuació obtinguda o asignada.
```

LabelType

Aquesta classe asigna al número 1 nivell POSITIU, al 0 nivell NEUTRE i al -1 NEGATIU.

Ontology

La classe Ontology serveix per guardar les classes semàntiques, o etiquetes que tenim.

```
Ontology()
```

Dins de l'ontologia tenim un set de classes.

Funcions

```
def doesSemanticClassExist( className ):
```

className és el nom d'una de les etiquetes posiblement coningudes a la ontologia.

Aquesta funció retorna un boleà indicant si className existeix dins del conjunt d'etiquetes, o no.

```
def addSemanticClass( className ):
```

Afegeix una etiqueta al conjunt d'etiquetes.

```
def getSemanticClassNames( ):
```

Retorna els etiquetes contingudes a la ontologia.

```
def saveOntology ( ontologyPathFile ):
```

ontologyPathFile: directori i nom del fitxer on es guardarà la ontologia

Desa a disc la ontologia.

```
def loadOntology ( ontologyPathFile ):
```

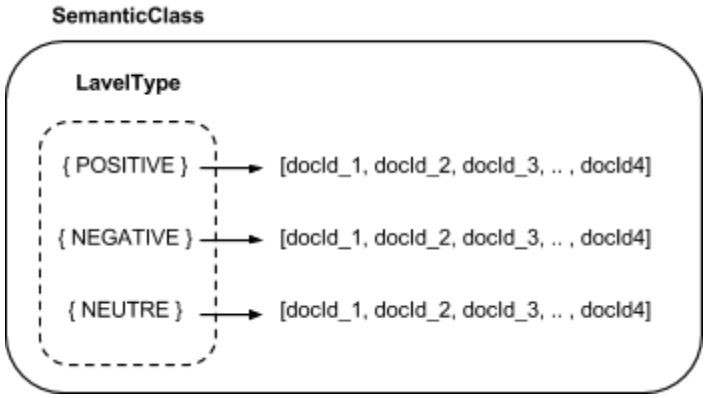
Carrega la ontologia desada al directori ontologyPathFile.

AnnotatedSemanticClass

Aquesta classe conté, per a una etiqueta semàntica específica, el nom de les imatges positives, negatives o neutres, organitzades als diferents diccionaris.

`AnnotatedSemanticClass()`

Conté un diccionari de nivells [POSITIU, NEGATIU, NEUTRE] com a keys i cada una té (o tindrà un cop omplert) una llista de les imatges que corresponen a cada nivell.



Funcions

`def addInstance(labelType, docId, score=1.0 ):`

labelType indica, per una etiqueta si la imatge pertany no pertany, o es neutral. Pot ser LabelType.POSITIVE, LabelType.NEUTRAL o LabelType.NEGATIVE

docId és el nom de la imatge.

score és la probabilitat de que sigui de la classe semàntica a la que s'esta afegint.

Afegeix una instància a la classe semàntica a la que ens estem referint.

`def getNofDocsInClass( labelType ):`

Retorna el nombre d'imatges que tenen el valor indicat amb el **labelType** (POSITIU, NEGATIU o NEUTRE) per a la classe semàntica a la que ens estem referint.

`def getDocIdsInClass( labelType ):`

Retorna els noms de les imatges que tenen el valor indicat amb el **labelType** per a la classe semàntica a la que ens estem referint.

`def containsDocId( labelType, docId ):`

Retorna verdader o fals si la imatge **docId** pertany al **labelType** que indiquem.

```
def addPos():
```

Afegeix una posició a la llista que hi ha dins de

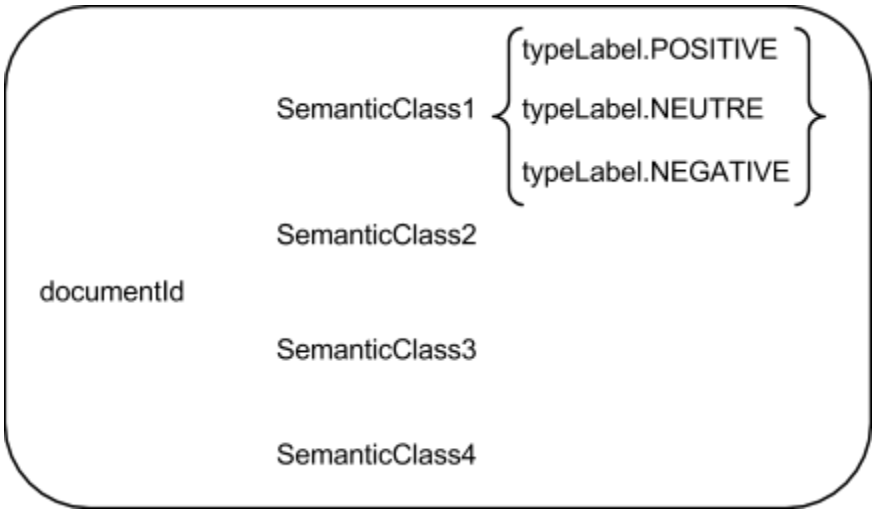
Annotation

Conté una col·lecció de imatges anotades semànticament en forma de diccionari.

```
Annotation(ontology)
```

Funcions

```
def isSemanticClassInstantiated( className ):
def getSemanticClassNames( ):
def addInstance( className, labelType, docId ):
def getNofDocsInClass( className, labelType ):
def getDocIdsInClass( className, labelType ):
def getAnnotatedClasses( labelType, docId ):
def completetheAnnotation( ):
```



Cada ítem de una anotació té el nom de la imatge anotada, les possibles classes semàntiques i el valor [POSITIU, NEGATIU o NEUTRE] que agafa cada una de les classes.

Trainer

La classe *Trainer* serveix per entrenar un classificador, donats un directoris facilita la carrega de vectors de característiques i genera un *Model* que permet desar a disc.

```
Trainer(pathDirFeatures, annotationFile, flagVerbose=False )
```

Funcions

```
def loadData( pathTxtFile ):
```

```
def run( pathTxtFile ):
```

```
def save_model_to_disk( savepath, model_name = 'model_svm.p'):
```

Detector

La classe Detector permet fer la detecció d'una o d'un conjunt de imatges, facilita carregar els descriptors i retorna els resultats obtinguts en un fitxer o en forma d'*Annotation*. També podem combinar els resultats obtinguts a partir de diferents fonts de característiques.

```
Detector ( Model , ontology)
```

Funcions

```
def get_labels( ):
def createCompleteAnnotation(  ):
def loadDescriptor( pathTxtFile, pathDirFeatures ):
def predict_collection( pathTxtFile,pathDirFeatures ):
def get_results( ):
def save_to_file(  pathFileResults):
def predict_class(  descriptor):
def combine_voting_in_added_with(   feature_voting):
def predict_class_from_voting( combinate_voting):
def combine_ponderating_voting_with( feature_voting,    ponderation):
def get_voting_collection( pathTxtFile,pathDirFeatures ):
def get_voting( descriptor):
def saveAnnotation (  AnnotationPath):
```

Features

BofExtractor

Extreu els descriptors de Bag of SIFT Features (BoF) de una imatge o de d'una colecció, també desa a disc el descriptor.

```
BofExtractor( pathVisualVocabulary,
              flagSaveInMemory=False,
              flagVerbose=False,
              flagRunOnServer=False):
```

Funcions

```
def processDir(  pathDirImages, pathDirBoFs=None ):
def processTxtFile(  pathTxtFile,
                     pathDirImages,
                     fileImageExtension,
                     pathDirBoFs=None):
def processCollectionFilesImage( imageIds,
                                 pathDirImages,
                                 fileImageExtension,
                                 pathDirBoFs=None ):
def processCollectionFilesImage( imageIds,
                                 pathDirImages,
                                 fileImageExtension,
                                 pathDirBoFs=None ):
def processImage( pathFileImage ):
def saveToDisk( signature, pathFileBof ):
def getBofs( ):
```

SiftExtractor

Extreu les característiques SIFT de una imatge o de d'una col·lecció, també desa a disc el descriptor.

```
SiftExtractor( flagSaveInMemory=False, flagVerbose=False):
```

Funcions

```
def processDir( pathDirImages, pathDirSifts=None ):
def processTxtFile( pathTxtFile,
                    pathDirImages,
                    fileImageExtension,
                    pathDirSifts=None):
def processCollectionFilesImage( imageIds,
                                 pathDirImages,
                                 fileImageExtension,
                                 pathDirSifts=None ):
def processImage(  pathFileImage ):
def saveToDisk( descriptors, pathDirSifts, imageId):
def getDescriptors( ):
```

TextVocabulary

Crea un vocabulari textual obtenint les paraules més freqüents d'un conjunt d'entrenament.

```
TextVocabulary(pathFile)
```

Funcions

```
def buildFromTags( pathFileTags, numOfTerms ):
def get_most_freq_tags( ):
def loadFromDisk( pathFile):
def saveToDisk( pathFile):
```

TfidfExtractor

Extreu les caracteristiques TFIDF de les metadades de una imatge o d'una colecció a partir d'un vocabulari visual. També permet desar les característiques a disc.

```
TfidfExtractor( pathMetadata,
                pathFileTags,
                pathFileVocabulary,
                flagSaveInMemory = False,
                flagVerbose=False):
```

Funcions

```
def feature_extractor(  imageId):
def processTxtFile(  pathTxtFile, pathDirTfidfs=None):
def processCollectionFilesData( imageIds, pathDirTfidf=None ):
def processData(  imageId):
def saveToDisk( descriptors, pathDirTfidfs, imageId):
def getDescriptors( ):
```

VisualVocabulary

Crea un vocabulari visual i quantitza les caracteristiques visuals.

```
VisualVocabulary(pathFile=None, flagVerbose=False ):
```

Funcions

```
def readImageIdsFromTxtFile( pathTxtFile ):
def buildFromImageCollection( pathTxtFile,
                              pathDirImages,
                             fileImageExtension,
                             vocabularySize=4096,
                             maxNumImages=sys.maxint ):
def buildFromImageCollectionWard( pathTxtFile,
                                    pathDirImages,
                                    fileImageExtension,
                                    vocabularySize,
                                    maxNumImages=sys.maxint ):
def loadFromDisk( pathFile):
def saveToDisk( pathFile):
def quantizeVector(  descriptors ):
```

Tools

Les tools del projecte son scripts fets per cridar a cada una de les classes esmentades anteriorment, executant una a una en l'ordre assignat pels números davant del nom de la tool fan funcionar pas a pas el sistema d'etiquetatge.

En aquest cas, les tools de classificació d'imatges es troben a l'adreça *socialevent/mediaeval2013/classification*

2_datasets.py

Per començar a treballar necessitem les dades en un format concret, per tal de generalitzar el procés.

De totes les imatges que tenim, les separem en dos directoris, train i test, d'aquesta manera queden diferenciades les dades que farem servir per desenvolupament de les que farem servir per fer el testeig del nostre classificador.

Per a cada partició creem un objecte de tipus Dataset, i cridem la funció membre Dataset.build, que ens generarà un fitxer amb extensió txt amb el nom de totes les imatges que hi ha en aquell directori.

Aquest pas previ fa que després el processament de les dades sigui molt més ràpid.

3_vocabulary

Aquest script et crea un vocabulari tant visual com de text. Per a la part visual el vocabulari es crea a partir de característiques SIFT. El vocabulari creat ens servirà per extreure les BoF de cada imatge. Hi ha diferents mètodes d'extracció de descriptors d'imatge, nosaltres fem servir el SIFT.

Des de la funció VisualVocabulary.buildFromImageCollection es crida a l'extracció de característiques SIFT, per tant per obtenir un vocabulari visual hem de cridar aquesta funció i ens guardem el vocabulari mitjançant la funció VisualVocabulary.saveToDisk.

Similar al vocabulari visual, en aquest cas tenim com a descriptor el TF-IDF, en aquest cas no seria necessari crear un vocabulari com a tal, però la gran quantitat de dades de text que tenim fa que sigui necessari crear un vocabulari més petit i concret.

Creem un objecte de tipus TextVocabulary i cridem la funció TextVocabulary.buildFromTags tot dient-li en número de paraules que volem agafar per fer el vocabulari i el guardem amb la funció TextVocabulary.saveToDisk.

Aquests vocabularis creats els necessitem per poder crear els vectors de característiques BoF (Bag of Features) i TF-IDF

4_bof

Aquesta tool crea un objecte de tipus BofExtractor i cridem la funció BofExtractor.processTxtFile, que mitjançant el fitxer de text creat al principi amb la classe Dataset, extreu el BoF de cada imatge.

S'ha de tenir en compte que les imatges que hem de fer servir per crear el vocabulari son les de entrenament, i aquest mateix vocabulari es fa servir per extreure els BoF de entrenament i de test.

4_tfidf

Aquesta tool serveix per crear els vectors de TF-IDF, per tant creem un objecte de tipus TfidfExtractor i cridem la funció TfidfExtractor.processTxtFile.

5_annotation

Aquest script et crea un objecte de tipus Ontology on hi guardarem la informació sobre les etiquetes semàntiques que tenim cridant la funció Ontology.addSemanticClass. Això es una cosa que només hem de fer un cop. Seguidament creem un objecte de tipus Annotation, al qual li passem l'objecte ontology, i per cada una de les imatges d'entrenament cridem la funció Annotation.addInstance que afegeix a l'anotació cada imatge amb l'etiqueta semàntica que li correspon. Aquest objecte conté la informació sobre a quina classe pertany cada imatge.

5_models

Per crear els models que ens serviran per fer la detecció, i s'obtenen entrenant el classificador. Hem de crear un objecte de tipus Trainer , li donarem la informació del Annotation i les extraccions de característiques. Per tal de crear un model hem de cridar a la funció Trainer.run, després guardem el model a disc amb la funció Trainer.save_model_to_disk i ja tindrem el model creat.

6_Detector

Aquest script permet la classificació d'imatges a partir del model obtingut al entrenament. Creem un objecte de tipus Detector, i cridem a la funció Detector.predict_collection, ens retornarà un objecte de tipus annotation amb el nom de cada imatge i la classe semàntica que ha assignat

7_evaluation

El script d'avaluació permet comprovar quant bo és l'etiquetat dut a terme. A partir de l'Annotation obtingut a la sortida del detector creem un vector on cada posició correspon a una imatge del dataset. Després, per avaluar la precisió i el record posem 1 a la posició que es correspon. Aquests dos vectors seran la entrada per a la funció precision_recall_curve de les funcions de 'metrics' de la llibreria sklearn.