Memòria

Intel·ligència Artificial

Marc Armengol

Hector Ribes

Aleix Aiguasenosa

Roger Calaf

Índex

Introducció

Una empresa ens demana elaborar un projecte, en el qual un software que retorni els noms dels colors predominants d'una imatge donada, és a dir, que donada una imatge retorni una llista de fins a 5 noms dels colors més importants entre els de la següent llista:
white, gray, black, red, green, blue, yellow, orange, brown, purple, pink .
Els colors han de representar un mínim d'un 5% del contingut de les imatges.

Dades psicofísiques (quins noms assigna la gent a un RGB)

R	G	B	White	Gray	Black	Red	Green	Blue	Yellow	Orange	Brown	Purple	pink
163	74	86	6.7	0	3	0	0	0	0.2	0.1	0	0	0
202	80	91	9.25	0.05	0.35	0	0	0	0.05	0.3	0	0	0
217	83	112	2.75	0	0	0	0	0	0.25	7	0	0	0
226	119	134	2.55	0	0	0	0	0	0	7.45	0	0	0
230	173	170	1.1	0.25	0.65	0	0	0	0	8	0	0	0
234	201	197	0.5	0.15	1.3	0	0	0	0	8.05	0	0	0
231	227	222	0	0	3.2	0.3	0.1	0	0	0	0.9	5.5	0
164	80	72	2.85	0.1	7.05	0	0	0	0	0	0	0	0
203	80	78	8.6	0	1.4	0	0	0	0	0	0	0	0
220	105	91	7.1	2.35	0.2	0	0	0	0	0.35	0	0	0
221	123	125	4.1	0.3	0	0	0	0	0	5.6	0	0	0
231	172	164	0.75	0.65	0.5	0	0	0	0	8.1	0	0	0
234	201	193	0	0.1	1.45	0	0	0	0	8.35	0	0.1	0
231	227	221	0	0.15	2.55	0.2	0.25	0	0	0	0.7	6.15	0
163	81	67	2.05	0	7.95	0	0	0	0	0	0	0	0
203	85	70	7.55	0.85	1.6	0	0	0	0	0	0	0	0
219	90	74	8.4	1.35	0.25	0	0	0	0	0	0	0	0
227	128	93	1.65	7.7	0.25	0	0	0	0	0.4	0	0	0
230	168	151	0.65	2.35	1.1	0	0	0	0	5.9	0	0	0
239	199	185	0	0.6	1.05	0	0	0	0	8.35	0	0	0

En aquesta memòria exposarem les parts principals del projecte, com hem desenvolupat el projecte, els principals problemes que hem tingut i com els hem resolt. També descriurem possibles millores del programa de manera que sigui mes eficient.

Per a realitzar el projecte farem servir un llenguatge de programació orientat a objectes anomenat Python. Utilitzarem Python perquè es un llenguatge molt potent i fàcil de programar ja que la seva sintaxis es molt intuïtiva.

En la part gràfica, farem servir el conjunt de llibreries anomenada PyQt. PyQt es un conjunt de llibreries de la biblioteca gràfica Qt per Python. Conte un conjunt d'objectes gràfics per a poder desenvolupar aplicació amb formularis d'una manera fàcil ja te molta ajuda i suport a Internet.

Per a la mostra de la imatge, hem utilitzat la llibreria d'imatge per Python, la PIL. Aquesta llibreria ens permet tractar la imatge pixel per pixel i aplicar modificacions i transformacions per aconseguir el que ens demanen.

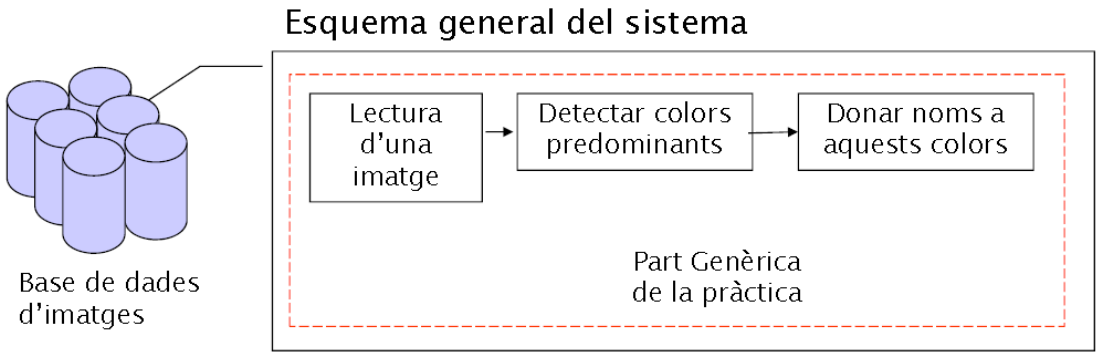
Objectius

Per tant, cal fer un software que retorni els noms dels colors predominants d'una imatge donada, és a dir, que donada una imatge retorni una llista de 4 o 5 noms dels colors més importants.

Caldrà doncs utilitzar un algorisme d'aprenentatge automàtic com es el K-means. A partir d'una imatge el programa ha de ser capaç de classificar els píxels de la imatge i decidir quins son els colors predominants.

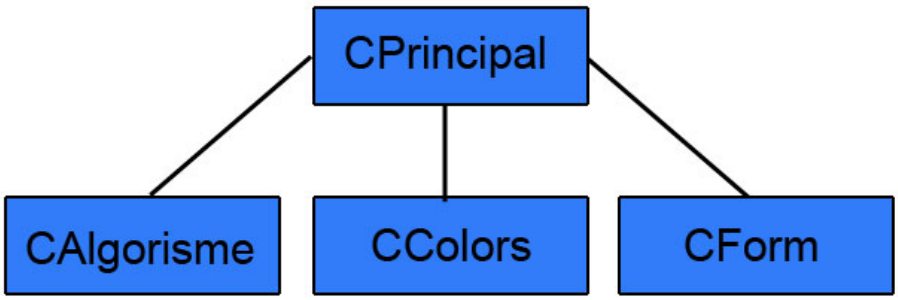
Especificació

L'esquema bàsic del programa serà el següent:



Per dur a terme aquest esquema, hem dividit el nostre programa en 3 submoduls. Un modul per contenir tota la part de l'algorisme K-means, un altre modul per implementar la part gràfica i un modul per determinar els colors de cada centre.

L'esquema modular de la practica es el següent:



ESQUEMA DEL PROGRAMA PRINCIPAL

1. Seleccionar imatge
2. Carregar pixels
3. Demanar la k
4. Aplicar K-means
5. Obtenir colors principals
6. Mostrar els colors per

Especificació dels mòduls

Modul CPrincipal()

Dades

llista_pixels : llista de pixels RGB
llista_colors : llista dels colors predominants
llista_centres : llista dels K centres de la imatge

Metodes

CPrincipal() : Constructor de la classe CPrincipal

carregapixels(fn) : Carrega els píxels de la imatge i els posa dintre la llista llista_pixels. El paràmetre fn es el path on es troba la imatge.

mostrar_imatge(fn): Carrega la imatge seleccionada i la mostra al formulari del programa. El paràmetre fn es el path on es troba la imatge.

navega(): Aquest mètode mostra un formulari per poder seleccionar la imatge al sistema de fitxers del sistema operatiu.

generar_centres(k): Aquest mètode, genera els centre de manera aleatòria per començar a aplicar l'algorisme K-means. El paràmetre k et diu el nombre de centres que te la imatge.

aplica_algoritme(): Mètode per aplicar l'algoritme K-means i trobar els 5 colors predominants de la imatge.

pintar_rectangles(): Aquest mètode mostra al formulari els 5 colors predominants de la imatge.

Modul CForm()

No especificarem els mòdul CForm perquè son un conjunt de mètodes que no son importants des de el punt de vista de desenvolupament de la practica. Nomes volem aclarar que aquest mòdul s'encarrega de la part gràfica del programa, conté un conjunt de mètodes per a poder mostrar el formulari del programa.

Modul CColors()

Metodes

CColors(): Constructor de la classe, en aquest cas anirà buit perquè no hem d'inicialitzar res.

lee(cadena): Aquest mètode passa una cadena de caracters a una llista.

lee_cadenas(cadena): Extreu els espais en blanc de les cadenes.

leer(fitxer): Llegeix el fitxer de colors i el retorna com una llista. El paràmetre fitxer es el descriptor de l'arxiu.

probabilitat(centre,colors): Aquest mètode calcula la distancia mínima que hi ha des de un centre al color de la taula.

ordena(x, y): Compara dos variables. Aquest mètode serveix ordenar la llista de colors segons el % de píxels que pertanyen a cada color.

diu_colors(centres): Aquest es el mètode principal de la classe. Com a paràmetre li passem la llista de centres obtinguda fent l'algorisme K-means. Aquesta funció ens retornarà la llista amb els centres, el nom del color que pertany aquell centre i el % de píxels.

Mòdul CAlgorisme()

Mètodes

Calcular_distancia(x,y): Passant com a parametres dos elements x,y. Et calcula la distancia entre ells.

Recalcular_centres(Con, Col, Cen): Aquesta funció, recalcula els centres un cop hem classificat els píxels. Quan recalculem els centres i veiem que els centres no es mouen de lloc llavors es quan hem de parar de classificar els píxels perquè ja els tenim ben classificats.

Comprovar_final(Cen1,Cen2): Comprova si dos centres son iguals o no. Com a paràmetres li passem les coordenades de 2 centres. La funció retorna cert o fals segons el que sigui el resultat.

K_means(Colors,Centres): Funció principal d'aquesta classe. Es la funció que aplica l'algorisme K-means. Com a paràmetres d'entrada li passem la llista amb tots els píxels de la imatge i una llista amb els k centres obtinguts aleatòriament. La funció ens retorna una llista amb la posició dels K centres.

Implementació de CAlgorisme

Nomes ens centrarem en la part important del codi, ja que l'objectiu important de la pràctica es la implementació de l'algorisme K-means. Per tant, en el nostre projecte la classe CAlgorisme es la que te implementada l'algorisme K-means. Per això, nomes ens centrarem en explicar com hem implementat aquesta classe. El pseudo-codi es el següent:

```
classe CAlgorisme():

    funció Calcular_distancia(x,y: llistes):
        z := []
        dist := 0
        Per i desde 0... longitud(x):
            z.insertar(x[i]-y[i])
            dist := dist + z[i]*z[i]
        FiPer
        retorna dist

   funcio  Recalcular_centres(Con, Col, Cen: llistes):
        Nous = []
        Per  x desde 0... longitud(Cen):
            Nous.insertar([0,0,0])
            CenR := 0
            CenG := 0
            CenB := 0
            count := 0
            i := 0

            Per  y desde 0..Con:
                 Si x=y :
                    CenR := CenR + Col[i][0]
                    CenG := CenG + Col[i][1]
                    CenB := CenB + Col[i][2]
                    count = count + 1
         Fsi
                 i := i + 1
            FiPer
            Si count != 0:
                Nous[x][0] := CenR/count
                Nous[x][1]: = CenG/count
                Nous[x][2] := CenB/count
      Fsi
        Fper
        retorna Nous
```

```
funcio Comprovar_final(Cen1,Cen2: llistes):
    trobat : = True
    Per x desde 0...longitud(Cen1):
        Si trobat:
            trobat := (Cen1[x]=Cen2[x])
       Fsi
    Fiper
   retorna trobat

funcio K_means(Colors,Centres: llistes):
    Contadors := []
    Centres2 : =[]
    Per i desde 0...longitud(Colors):
        Contadors.insertar(0)
    Fiper
    Per j desde 0...longitud(Centres):
        Centres2.insertar([0,0,0])
    Fiper
    final := False
    mentre (no final):
        Per i desde 0...longitud(Centres):
            Per j desde 0 ...3:
                Centres2[i][j] :=Centres[i][j]
    FiPer
        FiPer

        Per x desde 0...longitud(Colors):
            con := 0
            minim := 0
            dist := 0
            Per y desde 0...longitud(Centres):
                dist := Calcular_distancia(Colors[x],Centres[y])
                Si y=0:
                    minim := dist
               Sino:
                    Si dist < minim:
                        minim := dist
                        con := y
             Fsi
        Fsi
    FPer
            Contadors[x] := con
        Centres := Recalcular_centres(Contadors,Colors,Centres)
        final := Comprovar_final(Centres,Centres2)
    FPer
    Centres_i_pixels = [[]]
Fmentre
    Per x desde 0...longitud(Centres):
        Si x = 0:
            Centres_i_pixels[x].insertar(Centres[x])
        Sino:
            Centres_i_pixels.insertar([Centres[x]])
        Fsi
        cont := 0
        Per y desde 0 ... Contadors:
            Si y=x:
                cont :=  cont + 1
    Fsi
```

```
     FPer
    Centres_i_pixels[x].insertar(cont)
FPer
retorna Centres_i_pixels
```

Conclusions

El problema de l'aprenentatge automàtic, no es trivial. Hi ha molts condicions a tenir en compte. El programador ha de ser conscient que ha de fer que l'ordinador reconegui els colors a partir de classificar els píxels que te una imatge, no que a partir dels colors classifiqui els píxels, per això aquesta practica ens ha sigut molt difícil de desenvolupar. Vam haver d'orientar la practica d'una manera totalment diferent com la teníem plantejada. Tot i així hem pogut resoldre el problema i obtenir una solució mes o menys estable.

Uns dels principals problemes ha tenir en compte, per fer aquest tipus d'aplicacions son els tamanys de les imatges. Per una imatge gran, l'algorisme es lent i costos i s'han de buscar solucions alternatives.

Una solució alternativa seria l'ús de threads. D'aquesta manera repartiríem la feina de classificació de píxels per centre a threads diferents i d'aquesta manera reduiríem els temps d'execució del programa.

Bibliografia

URL: http://stackoverflow.com/questions/10477075/pyqt4-jpeg-jpg-unsupported-image-format
Data: 7/05
Títol: PyQt4 jpeg/jpg unsupported image format
Idioma: Angles
Descripció: Error en python.

URL: http://www.riverbankcomputing.co.uk/static/Docs/PyQt4/html/classes.html
Data:
Títol: PyQt Class Reference
Idioma: Angles
Descripció: Descripció de les classes i mètodes de les llibreries Qt.

URL: http://en.wikipedia.org/wiki/K-means_clustering
Data: K-means clustering
Idioma: Angles
Descripció: Explicació sobre l'algorisme K-means