FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Olhó-passarinho: uma extensão do TweeProfiles para fotografias

Ivo Filipe Valente Mota

VERSÃO DE TRABALHO

Mestrado Integrado em Engenharia Eletrotécnica e de Computadores

Orientador: Luís Filipe Pinto de Almeida Teixeira (PhD)

Co-orientador: Carlos Manuel Milheiro de Oliveira Pinto Soares (PhD)

30 de Junho de 2014

Conteúdo

Lista de Figuras

Lista de Tabelas

Abreviaturas e Símbolos

GoF/GOP	Grupo de *Frames* / Grupo de Imagens
SA	*Shape Adapted*
MS	*Maximally Stable*

Capítulo 1

Introdução

As redes sociais são uma excelente fonte de informação sempre em atualização, que fornece aos investigadores vasta quantidade e variedade de dados. Este dados apresentam-se de diferentes formas como textos, imagens e vídeos.

O Twitter é um serviço de microblogging, que permite aos utilizadores partilharem mensagens, designadas por *tweets*, até um máximo de 140 caracteres. O Twitter, ao contrário de outras redes sociais como o Facebook e Linkedin que utilizam uma rede de comunicação bi-direcional, esta utiliza uma infraestrutura assimétrica onde existem os *"friends"* e os *"followers"*. Supondo que é um utilizador do Twitter, os *"friends"* corresponde às contas das pessoas que o utilizador segue e os *"followers"* corresponde às contas das pessoas que o seguem [8].

O TweeProfiles [9], é uma ferramenta que recorre à rede social Twitter com o objetivo de identificar padrões em mensagens escritas em português partilhadas nesta rede social. Esta ferramenta utiliza processos de *Data Mining*, mais precisamente de *Text Mining*. A principal característica do TweeProfiles é o facto de utilizar a tarefa de *clustering* para identificar padrões, isto é, não se limita a apresentar *clusters* pelo conteúdo das mensagens (o texto), mas associando também as dimensões temporais e espaciais das mensagens.

1.1 Motivação

Para além do Twitter ser uma rede social que permite a partilha de *tweets*, este permite a partilha de imagens a partir do próprio serviço, ou através de outros serviços como Twitpic ou Flickr. Também, a partilha da ligação a uma imagem, é possível através do serviço Instagram, sendo que neste caso o acesso à imagem é redirecionado para o serviço Instagram. Devido ao grande número de utilizadores e de informação partilhada a todo o instante no Twitter, este torna-se um excelente serviço de pesquisa e análise, proporcionando aos investigadores e empresas uma quantidade e variedade de dados necessários para o desenvolvimento de ferramentas de extração de conhecimento e identificação de padrões.

1.2 Objetivos

Esta dissertação tem como principal objetivo a criação de uma extensão para o TweeProfiles através de técnicas de processamento de imagem e *Data Mining*, que permita a identificação de padrões em imagens partilhadas no serviço de microblogging Twitter, com representação de *clusters*.

Será assim necessário o desenvolvimento de um módulo responsável pela recolha das imagens através dos urls de tweets partilhados no Twitter. Para essa recolha será utilizada a plataforma TwitterEcho [10], que consiste num projeto open source, responsável por extrair e armazenar tweets de uma determinada comunidade de utilizadores, tendo sido desenvolvido com o intuito de ajudar os investigadores a terem facilidade de acesso a uma base de dados de tweets, na sua maioria, escritos na língua portuguesa. Posteriormente, deverá ser desenvolvido um módulo que utilize ferramentas de processamento de imagem para extração e análise da informação das imagens, e através da tarefa de *clustering*, seja capaz de apresentar *clusters* de imagens. Por fim, deverá ser integrado na ferramenta TweeProfiles, com objetivo de visualizar os *clusters* nas diferentes dimensões, como temporal, espacial e pelo conteúdo das imagens.

1.3 Estrutura do documento

O documento está organizado da seguinte forma: o capítulo 2 reflete o estado da arte, onde é apresentado uma pesquisa sobre os vários domínios científicos relacionados com as necessidades para o desenvolvimento do projeto de dissertação. Na secção 2.1 é descrito as características da tarefa de *clustering* do estudo e na secção 2.3, é referenciado as várias formas de representação computacional de imagens. Por fim, no capítulo 3, é referido o plano de trabalho para o desenvolvimento do projeto de dissertação, as ferramentas necessárias e as conclusões retiradas.

Capítulo 2

Conceitos e trabalhos relacionados

Neste capítulo é apresentado o estudo preliminar realizado, tendo em vista a aquisição de competências e conhecimentos necessários para o desenvolvimento do projeto, que se focam essencialmente em análise de métodos de *data mining* e em técnicas aplicadas em visão por computador. Em primeiro ligar será exposto conteúdo relativamente a métodos de *clustering* como uma tarefa de *data mining*. Em seguida será apresentado formas de representação de imagens. Por fim, são referenciados alguns trabalhos relacionados com o projeto a desenvolver nesta dissertação.

2.1 Clustering

Extração de conhecimento em base de dados ou *Data Mining* é um processo de exploração de grande quantidades de dados que procura encontrar padrões interessantes [7]. Trata-te assim de uma fusão de estatística aplicada, sistemas de lógica, inteligência artificial, *Machine Learning* e gestão de base de dados [11]. Este processo é caracterizado por várias tarefas possíveis de ser aplicadas, dependendo do problema abordado, tais como [12]:

- Deteção de anomalias (outliers/ alterações/ desvios) - Identificação de registos de dados incomuns, podendo ser erros nos dados ou objetos interessantes que apresentam comportamento diferente dos restantes;
- Regras de associação - Procura relações entre variáveis que ocorram frequentemente;
- Classificação - É a tarefa de generalizar uma estrutura conhecida e aplicar a novos dados, sendo essencialmente utilizada em tarefas de previsão;
- Regressão - Tenta encontrar uma função que se modela os dados com o mínimo de erro;
- Resumo - Trata-se da representação mais compacta do conjunto de dados, que pode incluir visualização e descrição através de um relatório,
- *Clustering* - Tarefa de descobrir grupos em que os dados apresentam de alguma forma semelhanças, sem o uso de estruturas previamente conhecidas

Este projeto terá como principal tarefa a realização de *clustering*. Pode-se definir *clustering* como "um processo de agrupamento de um conjunto de objetos de dados em vários grupos ou *clusters*, de modo que os objetos dentro de um *cluster* apresentem alta similaridade, mas que sejam muito diferentes de objetos de outros *clusters*. Diferenças e semelhanças são avaliados com base nos valores de atributos que descrevem os objetos e muitas vezes envolvem medidas de distância" [7].

A realização de *clustering* é assim uma escolha lógica para a extração de padrões em dados não supervisionados e para o agrupamento de *tweets* pela sua semelhança em conteúdo, neste caso as imagens, e com a integração de outras dimensões, como o tempo e espaço.

O *clustering* faz parte de um conjunto de técnicas aplicadas na aprendizagem não supervisionada. Enquanto que na aprendizagem supervisionada, existe um conjunto de dados previamente analisados e rotulados que são usados para treinar um modelo para que seja capaz de encontrar relação entre os atributos desse dados com novos conjuntos dados, na aprendizagem não supervisionada, não são utilizados conjuntos de dados previamente analisados e rotulados. Assim o processo de descoberta de padrões nos dados apenas tem em conta os dados presentes, tentando organizar as instâncias em grupos semelhantes [13].

Nesta secção serão apresentadas as principais características e técnicas para aplicação da tarefa de *clustering*, tais como, *clustering* por partição, *clustering* hierárquico, *clustering* baseado em densidade, *clustering* baseado em grelhas, funções de distância para o cálculo da similaridade entre objetos e por fim, a avaliação de *clusters*.

2.1.1 Clustering por partição

A utilização de métodos baseados em partições é a forma mais simples e elementar de realizar análise por *clustering*, em que um conjunto de objetos é distribuído em vários grupos ou *clusters* mais pequenos. É assumido que o número de *clusters* é conhecido antes da realização da tarefa, sendo esse valor tomado como o ponto de partida para aplicação de métodos baseados em partição [7].

2.1.1.1 Algoritmo K-Means

O algoritmo *k-means* é o melhor algoritmo de *clustering* por partição e o mais utilizado devido à sua simplicidade e eficiência [13]. É apresentado como sendo um algoritmo de *clustering* por partição, pois este divide o conjunto de dados em partições mais pequenas, formando assim os *clusters*.

Inicialmente é necessário que o utilizador indique o valor de k e o algoritmo irá iterativamente dividir o conjunto de objetos em k-*clusters* diferentes, baseado em funções de distância [13] que são apresentadas na subsecção 2.1.5.

Cada *cluster* apresenta um centroide que é o representante do grupo, sendo o valor médio de todos os objetos (instâncias) pertencentes ao *cluster*. Este centroide é recalculado de forma

iterativa até que seja atingido o critério de paragem. A convergência ou critério de paragem pode ser um dos seguintes enumerados:

1. Não ocorre (ou ocorre um valor mínimo) de alterações dos objetos para diferentes *clusters*.
2. Não ocorre (ou ocorre um valor mínimo) de alterações dos centroides.
3. Diminuição mínima da **soma do erro quadrático** (SEQ),

$$SEQ = \sum_{j=1}^{k} \sum_{x \in C_j} dist(x, m_j)^2, \tag{2.1}$$

onde k é o número de *clusters* pretendidos, C_j é i-íssimo *cluster*, m_j é o centroide do *cluster* C_j e $dist(x, m_j)$ é a distância entre uma instância **x** e o centroide m_j.

Assim, "o algoritmo *k-means* pode ser usado em qualquer aplicação com um conjunto de dados onde a média pode ser definida e calculada" [13].

No **espaço euclidiano**, o valor médio (centroide) de um *cluster* é calculada da seguinte forma:

$$m_j = \frac{1}{|C_j|} \sum_{x_i \in C_j} x_i, \tag{2.2}$$

onde $|C_j|$ é o número de pontos (instâncias) no *cluster* C_j. A distância entre um ponto x_i a um centroide m_j é calculado da seguinte forma:

$$dist(x_i, m_j) = ||x_i - m_j|| = \sqrt{(x_{i1} - m_{j1})^2 + (x_{i2} - m_{j2})^2 + ... + (x_{ir} - m_{jr})^2}. \tag{2.3}$$

O pseudo-código deste algoritmo é apresentado no algoritmo 1.

Algorithm 1 K-Means

```
1: procedure K-MEANS(k: clusters, D: conjunto de dados)
2:     escolher k objetos de D como centroides dos clusters iniciais;
3:     repeat
4:         (re) atribuir cada objeto a ao cluster ao qual o objeto é o mais similar;
5:         actualizar o centroide do cluster;
6:     until clusters sem alterações;
7:     return conjunto de k clusters;
8: end procedure
```

2.1.2 Clustering hierárquico

O *clustering* hierárquico é outra abordagem importante na tarefa de *clustering*. Os *clusters* são criados sobre a forma de uma sequência em árvore (dendrograma). Os objetos (instâncias) encontram-se no fundo do diagrama, enquanto que o conjunto de todos os objetos encontra-se no topo do diagrama. Cada nó que se encontra no interior do diagrama possui nós filhos, sendo que cada nó representa um *cluster*. Assim designa-se por *clusters* irmãos, aqueles que derivam

de um mesmo *cluster*, isto é, do nó parente [13]. A figura 2.1 exemplifica a representação de um dendrograma.

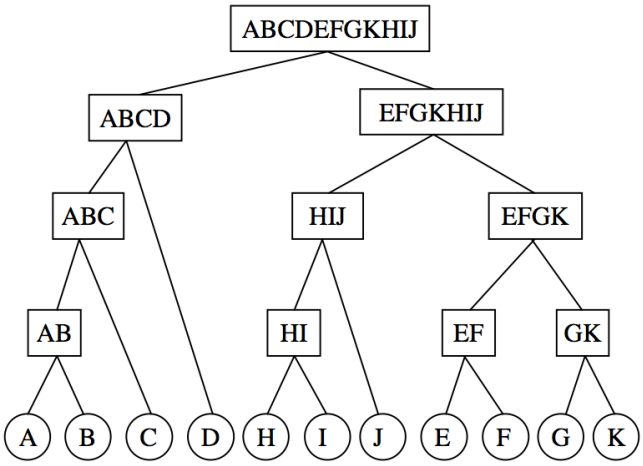


Figura 2.1: Dendrograma [1]

Existem dois tipos principais de métodos de *clustering* hierárquico, sendo eles [13] :

***Clustering* por aglomeração:** O dendrograma é construido do nível mais baixo até ao mais alto, juntando sucessivamente e iterativamente os *clusters* com maiores semelhanças até existir um único *cluster* com todo o conjunto dos dados.

***Clustering* por divisão:** O dendograma é construido do nível mais alto até ao nível mais baixo, onde o processo tem inicio com um único *cluster* que possui todos os objetos, sendo dividido sucessivamente em *clusters* mais pequenos, até que estes sejam constituídos apenas por um único objeto.

Ao contrário do algoritmo *k-means*, que apenas calcula a distância entre os centroides de cada grupo ou *cluster*, no *clustering* hierárquico pode ser usado os vários métodos apresentados em seguida para determinar a distância entre dois *clusters* [13]:

Método *Single-Link*: Neste método, a distância entre dois *clusters* é determinada pela distância entre os dois objetos mais próximos (vizinhos mais próximo) pertencentes a *clusters* diferentes.

Método *Complete-Link*: Neste método, a distância entre dois *clusters* é determinada pela maior distância entre dois objetos (vizinhos mais distante).

Método Average-Link: Este método tenta manter um compromisso entre a sensibilidade a *outliers* do método *Complete-Link* e a sensibilidade do método *Single-Link* ao ruído existente nos dados. Para isso, é determinada a distância entre dois *clusters* através da distância média entre todos os pares de objetos nos dois *clusters*.

Método Ward: Este método, tenta minimizar a variância entre dois *clusters* unidos.

Assim conclui-se que o *clustering* hierárquico apresenta-se para determinados domínios, como bastante intuitivo para humanos, mas a interpretação dos resultados pode ser por vezes subjetiva. Outra característica interessante é o facto de, ao contrário do *clustering* por partição, no *clustering* hierárquico não ser necessário especificar logo à partida o numero de *clusters*.

2.1.3 Clustering baseado em densidade

Os métodos de *clustering* por partição ou hierárquico estão preparados para encontrar *clusters* que apresentam formas geométricas circulares, sendo ineficiente quando as formas destes grupos são por exemplo elípticas. Assim, para descobrir *clusters* com formas arbitrárias, pode ser usado métodos baseados na densidade dos objetos [7]. Um dos algoritmos mais conhecidos que utiliza este tipo de método é o DBSCAN (*Density-Based Spatial Clustering of Applications with Noise*) que é capaz de encontrar *clusters* através da análise da densidade e proximidade dos objetos pertencentes a um conjunto de dados. Para esta análise é necessário previamente atribuir um valor que definirá o raio da vizinhança considerada para cada objeto. Esse parâmetro é designado por ε e terá de ser necessariamente maior que 0. Assim, a ε-vizinhança de um objeto x é o espaço dentro de um raio com valor ε, centrado em x [7]. Já para determinar a densidade de uma vizinhança, é utilizado o parâmetro *MinPts* também previamente definido, que especifica o número mínimo de objetos vizinhos que um objeto necessita ter em ser redor, para ser considerado como objeto central. Os passos necessários para a implementação deste método são apresentados no algoritmo 2

Uma das grandes vantagens do *clustering* baseado em densidade é que neste não é necessário uma previa definição do número de *clusters*, sendo apresentados os que encontrar consoante os dados que possui e os parâmetros definidos.

2.1.4 Clustering baseado em grelhas

Os métodos de *clustering* discutidos até agora, apresentam algoritmos que se adaptam a distribuição dos dados no espaço. Em alternativa, o *clustering* baseado em grelhas é orientado ao espaço, na medida em que divide o espaço em células independentemente da distribuição dos objetos de entrada. Este quantifica o espaço num número finito de células, que formam uma estrutura de grelhas sobre a qual são executadas as operações de *clustering*. Este método apresenta como principal vantagem, o tempo baixo de processamento, que normalmente é independentemente da quantidade de dados, no entanto, este depende do número de células em cada uma das dimensões no espaço quantizado [7].

O algoritmo STING (*Statistical Information Grid*) [14] é um dos algoritmos utilizados para *clustering* baseado em grelhas. Este divide o espaço em células retangulares, correspondente a diferentes resoluções que forma uma estrutura hierárquica, sendo a base o nível 1, os filhos o nível 2, e assim sucessivamente. Cada célula pertencente a um nível superior é dividida para formar células de menor dimensão no nível inferior seguinte. Assim, sabe-se que o nível mais baixo apresenta

Algorithm 2 DBSCAN

```
1:  procedure DBSCAN(MinPts: limiar da vizinhança, D: conjunto de dados, ε : raio )
2:      Marcar todos os objetos como não selecionados;
3:      repeat
4:          escolher aleatóriamente um objeto p não selecionado;
5:          actualizar objeto p como selecionado;
6:          if o ε-vizinhança de p tem pelo menos MinPts objetos then
7:              criar um novo cluster C e adicionar objeto p ao cluster C;
8:              Seja N o conjunto de objetos na ε-vizinhança de p;
9:              for cada ponto p' em N do
10:                 if p' não selecionado then
11:                     Marcar p' como selecionado
12:                     if a ε-vizinhança de p' tem pelo menos MinPts then
13:                         adicionar ponto a N
14:                     end if
15:                 end if
16:                 if p' não pertence a nenhum cluster then
17:                     adicionar p' a C
18:                 end if
19:             end for
20:             output C;
21:         else marcar p como ruido
22:         end if
23:     until todos os objetos selecionados;
24: end procedure
```

uma maior resolução. Isto permite que os *clusters* sejam encontrados recorrendo a uma pesquisa de cima para baixo (*clustering* por divisão, como explicado na sub-secção 2.1.2), passando por cada nível até atingir o mais baixo, retornando no fim as células mais relevantes para a consulta especificada. A informação estatística de cada célula é calculada e armazenada para o processamento de consultas futuras. Também é necessário ter em atenção que apenas é considerado para este algoritmo um espaço bidimensional. Um outro algoritmo com características semelhantes é o CLIQUE [15], que "identifica *clusters* densos em sub-espaços de máxima dimensão", isto é, são detetados todos os *clusters* em todos sub-espaços existentes e em que um ponto pode pertencer a vários *clusters* em sub-espaços diferentes.

2.1.5 Funções de distância

As funções de distância ou similaridade têm um papel fulcral em todos os algoritmos de *Clustering*. Existem enumeras funções de distância usadas para diferentes tipos de atributos (ou variáveis) [13]. Em seguida será apresentado diferentes funções distância para diferentes atributos como, numéricos, binários e nominal. Também será apresentado funções distância utilizados para as dimensões temporal e de conteúdo.

2.1.5.1 Atributos numéricos

As funções de distância mais utilizadas para variáveis numéricas são a **Distância Euclidiana** e **Distância Manhattan**. É utilizado $dist(x_i, x_j)$ para representar a distância entre duas instância de r dimensões. Ambas funções referidas anteriormente são casos especiais da função mais geral chamada **Distância Minkowski** [13]:

$$dist(x_i, x_j) = (|x_{i1} - x_{j1}|^h + |x_{i2} - x_{j2}|^h + ... + |x_{ir} - x_{jr}|^h)^{\frac{1}{h}}, \tag{2.4}$$

onde h é um inteiro positivo.

Se h=2, temos a **Distância Euclidiana**,

$$dist(x_i, x_j) = \sqrt{(x_{i1} - x_{j1})^2 + (x_{i2} - x_{j2})^2 + ... + (x_{ir} - x_{jr})^2}. \tag{2.5}$$

Se h=1, temos a **Distância City-block (Manhattan)**,

$$dist(x_i, x_j) = |x_{i1} - x_{j1}| + |x_{i2} - x_{j2}| + ... + |x_{ir} - x_{jr}|. \tag{2.6}$$

Também, não menos importantes, são outras funções distância apresentadas em seguida:

Distância Euclidiana Ponderada : A ponderação é atribuída através de pesos pela à importância que cada atributo representa relativamente a outros atributos.

$$dist(x_i, x_j) = \sqrt{w_1(x_{i1} - x_{j1})^2 + w_2(x_{i2} - x_{j2})^2 + ... + w_r(x_{ir} - x_{jr})^2}. \tag{2.7}$$

Distância Euclidiana Quadrática : Trata-se de uma alteração da função **Distância Euclidiana**, elevando a mesma ao quadrado, o que faz com que seja progressivamente atribuído peso maior a pontos dos dados que estejam mais afastados.

$$dist(x_i, x_j) = (x_{i1} - x_{j1})^2 + (x_{i2} - x_{j2})^2 + ... + (x_{ir} - x_{jr})^2. \tag{2.8}$$

Distância Chebychev : Utilizada para casos em que há necessidade de definir dois pontos dos dados como diferentes, caso sejam diferentes em qualquer dimensão.

$$dist(x_i, x_j) = max(|x_{i1} - x_{j1}| + |x_{i2} - x_{j2}| + ... + |x_{ir} - x_{jr}|). \tag{2.9}$$

2.1.5.2 Atributos binários e nominais

As funções apresentadas anterior apenas podem ser utilizadas com atributos do tipo numérico, assim será agora apresentado as funções de distância especificas para atributos do tipo binário e nominal.

Uma variável binária é aquela que apenas pode assumir dois estados ou valores, sendo normalmente representado pelo valor 0 e 1. Mas estes estados não apresentam um ordem definida.

Por exemplo, o caso de uma lâmpada, esta pode assumir apenas dois estados, ligado ou desligado, ou o género de uma pessoa, masculino ou feminino. Estes exemplos apresentam dois valores diferentes mas que não possuem qualquer ordem. As funções distância existentes para atributos binários são baseadas na proporção, sendo que a melhor maneira representar é através de uma matriz confusão [13].

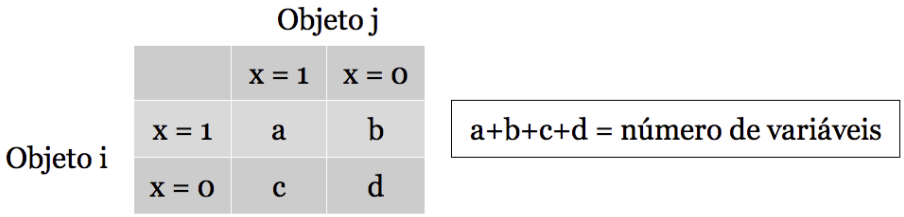


Figura 2.2: Matriz confusão de dois objetos com atributos binários

Os atributos binários ainda podem ser divididos em dois tipos de atributos diferentes, os simétricos e os assimétricos, sendo em seguido apresentado as funções distância para ambos oa casos [13].

Atributos simétricos: Um atributo é simétrico quando ambos os estados (0 ou 1) têm a mesma importância e o mesmo peso, tal como ocorre no exemplo dado anterior com o atributo género (masculino e feminino). Para este caso, a função distância mais utilizada é designada por *simple matching distance*, que corresponde à proporção de incompatibilidade ou desacordo(equação 2.10).

$$dist(x_i, x_j) = \frac{b+c}{a+b+c+d} \tag{2.10}$$

Atributos assimétricos: Um atributo é assimétrico se um dos estados apresenta maior importância ou valor do que o outro. Normalmente o estado mais valioso é o que ocorre com menor frequência. No nosso caso iremos considerar o estado 1 como o mais valioso. Assim, a função distância mais frequentemente utilizada para atributos assimétricos é a *Jaccard distance*:

$$dist(x_i, x_j) = \frac{b+c}{a+b+c} \tag{2.11}$$

No caso de atributos nominais com mais de dois estados ou valores, a função distância mais utilizada, é baseada na *simple matching distance*. Dado dois objetos i e j, r corresponde ao número

total de atributos e o q ao número de valores que são mutuamente correspondidos entre os objetos i e j:

$$dist(x_i, x_j) = \frac{r+q}{r} \tag{2.12}$$

2.1.5.3 Dimensão Temporal

O tempo é representado apenas por uma dimensão, sendo que para calcular a distância, por exemplo, entre dois tweets t_i e t_j, apenas é necessário calcular a diferença dos tempos entre os mesmos. Supondo que os valores dos tempos são respetivamente Δ_i e Δ_j, o intervalo de tempo pode ser definido pela seguinte equação:

$$dist^T(t_i, t_j) = |\Delta_i - \Delta_j| \tag{2.13}$$

Sendo que, para a dimensão temporal, também é possível utilizar a função distância euclidiana (equação 2.5).

2.1.5.4 Dimensão Espacial

Ao contrário da dimensão temporal, a dimensão espacial apresenta mais do que uma dimensão, latitude e longitude. Estas apresentam-se sobre a forma numérica, sendo possível o cálculo da distância entre dois objetos através de funções de distância para atributos numéricos como referido anteriormente (2.1.5.1). Assim, para o cálculo entre pontos distribuídos num espaço poderá-se-á recorrer à função Minkowski (equação 2.4), à função Euclidiana (equação2.5, à função Manhattan (equação 2.6) ou mesmo à função de Chebychev (equação 2.9), sendo que no caso mais específico de uma distribuição espacial geográfica, em que os pontos possuem latitude e longitude, é considerada mais apropriada a utilização da função distância Haversine (equação 2.14) pois esta toma em consideração a forma esférica da Terra. Obtendo um par de objetos x_i e x_j distanciados geograficamente, é considerado as latitudes ϕ_{x_i} e ϕ_{x_j} e longitudes λ_{x_i} e λ_{x_j} para determinar a distância através da seguinte equação,

$$dist^{Sp}(x_i, x_j) = 2R\sin^{-1}\left(\left[\sin^2\left(\frac{\phi_{x_i} - \phi_{x_j}}{2}\right) + \cos\phi_{x_i}\cos\phi_{x_j}\sin^2\left(\frac{\lambda_{x_i} - \lambda_{x_j}}{2}\right)\right]^{0.5}\right) \tag{2.14}$$

onde R representa o raio da Terra e que determina as unidades do resultado retornado pela função, sendo comum a utilização das unidades no sistema internacional (SI), o metro, podendo também ser representado em quilómetros devido ao fator de escala.

2.2 TweeProfiles

Esta dissertação pretende dar continuidade e um trabalho designado por TweeProfiles. O TweetProfiles é uma ferramenta de análise e visualização espaço-temporal de dados recolhidos no Twitter. Esta secção faz um breve introdução e descrição sobre esta ferramenta.

2.2.1 Objetivos

2.2.2 Descrição

2.2.3 Resultados ilustrativos

2.2.4 Prós e contras

2.3 Representação de informação visual

Na secção 2.1 foi apresentado o conceito e características da tarefa de *clustering* de uma forma geral. Nesta secção será exposto formas de representar imagens como dados. É importante salientar que a tarefa de *clustering* em imagem é muitas vezes associada à técnica de segmentação de imagem [16], não sendo este o objetivo deste projeto de dissertação, onde se pretende que para tarefa de *clustering* cada objeto seja considerado a imagem no seu conjunto.

2.3.1 Representação matricial

Uma imagem pode ser vista como um objeto (ou instância), sendo computacionalmente representada como uma matriz (um vetor bi-dimensional) de pixels. A matriz de pixels descreve assim a imagem como N x M m-bit pixels, onde N corresponde ao número de pontos ao longo do eixo horizontal, M o número de pontos ao longo do eixo vertical e m o número de bits por pixel que controla os níveis de brilho. Com m bits temos uma gama de valores para o brilho de 2^m, que varia entre 0 e $2^m - 1$. Assim se o valor de m for 8, os valores de brilho de cada pixel de uma imagem podem variar entre 0 e 255, que normalmente correspondem ao preto e branco respetivamente, sendo que os valores intermédios correspondem ao tons de cinza [17].

No caso de imagens a cores, o principio é idêntico, no entanto ao invés de se usar apenas um plano, as imagens a cores são representadas por 3 componentes de intensidade, designado por modelo *RGB*, a que corresponde respetivamente às cores vermelho (Red), verde (Green) e azul (Blue). Para além deste esquema de cores, também existe outros como o CMYK composto pelas componentes de cor, azul turquesa, magenta, amarelo e preto. Usando qualquer esquema de cores, existem 2 métodos principais para representar a cor do pixel. No primeiro método é utilizado um valor inteiro para cada pixel, sendo esse valor como um índice para uma tabela, também conhecida como palete da imagem, com a correspondência à intensidade de cada componente de cor. Este método tem como vantagem o facto de ser eficiente na utilização da memória, pois apenas é guardado um plano da imagem (os índices) e a palete (tabela). Por outro lado, tem como desvantagem o facto de normalmente ser usado um conjunto reduzido de cores o que provoca uma

redução da qualidade da imagem. Já o segundo método consiste na utilização de vários planos da imagem para armazenar a componente de cor de cada pixel. Este representa a imagem com mais precisão pois considera muito mais cores. Formato mais usual é 8 bits para cada uma das 3 componentes, no caso do RGB. Assim, são utilizados 24 bit para representar a cor de cada pixels, o que permite que uma imagem possa conter mais de 16 milhões de cores simultaneamente. Como era de esperar, isto envolve um custo grande na utilização de memória, mas a constante redução do custo das memórias faz com que este seja uma boa alternativa à apresentada anteriormente [17].

2.3.2 Histogramas

Outras das formas de representar a informação de uma imagem é através de um histograma. Um histograma de uma imagem apresenta a frequência de ocorrência de níveis individuais de brilho, representado através um gráfico que mostra o número de pixels da imagens com um determinado nível de brilho. No caso de pixels representados por 8-bit, o brilho vai variar de 0 (preto) até 255 (branco) [17]. Também pode ser apresentado informação de cor sobre uma imagem através de um histograma, sendo para isso necessário apresentar 3 histogramas diferenciados, um para cada componente de cor, no caso do esquema RGB. A figura 2.3 apresenta um exemplo de um histograma de uma imagem com tons cinza, onde são representados o número de pixeis para cada nível diferente de cinzento.

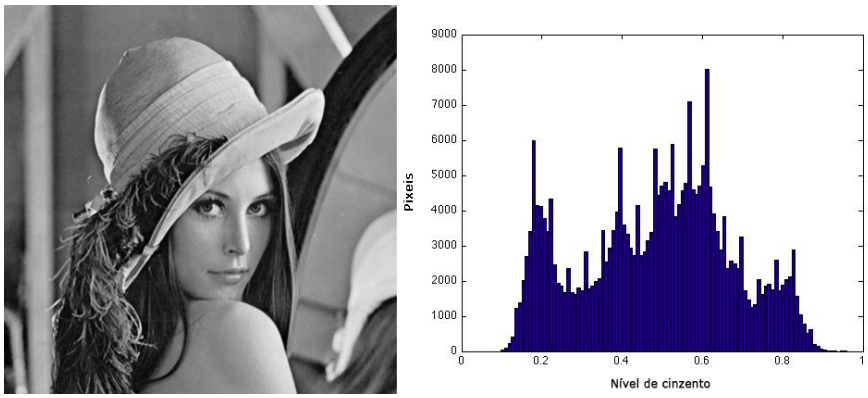


Figura 2.3: Imagem em tons cinza e respetivo histograma

2.3.3 Descritores de cor

A cor apresenta-se como um importante atributo da imagem para o olho humano e processamento por computador. Nesta subsecção são apresentados vários descritores de cor, utilizados para extração de informação e reconhecimento de similaridade em imagens. Por exemplo, o histograma de cores, referido na subsecção anterior 2.3.2, é um dos descritores de cor mais utilizados para caracterizar a distribuição da cor de uma imagem, mas apresenta uma baixa eficiência. Assim, em seguida é apresentado descritores de cor considerados pelo MPEG-7 [18, 19, 20, 21], que apresentam eficiência superior aos histogramas de cor.

2.3.3.1 Espaços de cor

Nesta subsecção é apresentado os vários descritores de espaços de cor especificado no MPEG-7 [21]. Existe uma vasta seleção de espaços de cores, tais como, RGB, YCbCr, HSV, HMMS, Monocromático e Matriz linear de transformação com referência a RGB. Estes são usados por outro descritor de cor, mais especificamente, o descritor de dominância de cor que será falado posteriormente. É utilizado também, um sinalizador para indicar a referência a uma cor primária e de mapeamento de um valor de referência do branco padrão.

Em espaços de cor, as componentes de cor são definidas como entidades de valor continuo, sendo que podem ser representadas por valores discretos através de uma quantização uniforme, em que é especificado um número de níveis de quantização para cada componente de cor no espaço de cor. A única exceção é o espaço de cor HMMD.

O espaço de cor RGB é um dos modelos referidos mais utilizados, que apresenta três componentes distintas, vermelho, verde e azul, tal como foi referido no subsecção 2.3.1. Neste modelo é utilizado a combinação das 3 cores primárias para representar as diferentes cores. O modelo YCbCr provém do padrão MPEG-1/2/4 [21] e é definido pela transformação linear do espaço de cor RGB como demonstrado na equação 2.15:

$$\begin{aligned} Y &= 0.299 \times R + 0.587 \times G + 0.114 \times B \\ Cb &= -0.169 \times R - 0.331 \times G + 0.500 \times B \\ Cr &= 0.500 \times R - 0.419 \times G - 0.081 \times B \end{aligned} \tag{2.15}$$

Para o espaço de cor Monocromático, é usado apenas a componente Y do modelo YCbCr.

O espaço de cor HSV apresenta uma especificação mais complexa, tendo sido desenvolvido para fornecer uma representação mais intuitiva e para se aproximar mais do sistema visual humano. A transformação do modelo RGB para o HSV não é linear, mas é reversível [18]. Uma das componentes é a matiz (H - *Hue*), que representa a componente de cor espectral dominante na sua forma mais pura, como o verde, amarelo, azul e vermelho. Ao ser adicionado branco à cor, esta sofre uma alteração, sendo que, adicionando mais branco, menos saturada se torna a cor. A saturação (S - *Saturation*) é precisamente outra das componentes deste modelo. Por fim, o valor (V - *Value*) corresponde ao brilho de cor.

O espaço de cor HMMD (*Hue-Max-Min-Diff*) [18, 21] é mais recente, que é caracterizado pela componente matiz, tal como o modelo HSV, pelo *max* e *min*, que são respetivamente o máximo e mínimo entre os valores R, G e B. Para descrever este modelo, também é utilizado a componente *Diff*, que corresponde à diferença entre o *max* e *min*. Para representar este espaço de cor, apenas é necessário três dos quatro componentes referidos anteriormente, como por exemplo, *Hue, Max, Min* ou *Hue, Diff, Sum*, onde *Sum* pode ser definida pela equação 2.16.

$$Sum = \frac{Max + Min}{2} \tag{2.16}$$

2.3.3.2 Cor dominante

O descritor de cor dominante fornece uma compacta representação das cores de uma imagem ou da região da imagem. Este apresenta a distribuição das cores mais representativas na imagem. Ao contrário do descritor de cor por histograma, na especificação do descritor de cor dominante, as cores mais representativas são calculadas a partir de cada imagem, em vez de ser fixado no espaço de cor, permitindo assim, uma representação das cores mais exatas e compacta, presentes numa região de interesse.

O descritor de cor dominante pode ser definido como,

$$F = c_i, p_i, v_i, s, (i = 1, 2, ..., N)$$

onde N é o número de cores dominantes. Cada valor c_i da cor dominante é um vetor de valores das componentes do espaço de cor correspondente (por exemplo, um vetor de 3 dimensões no espaço de cor RGB). O valor p_i é a fração de pixels na imagem ou região da imagem (normalizado para um valor entre 0 e 1) que corresponde à cor c_i, sendo $\sum_i p_i = 1$. O opcional v_i descreve a variação dos valores de cor dos pixels em um *cluster* em torno da cor representativa correspondente. Por fim, a coerência espacial s é um único número representa a homogeneidade espacial global das cores predominantes na imagem [21].

2.3.3.3 Cor escalável

O descritor de cor escalável, pode ser interpretado como um esquema de codificação base que recorre à transformada de Haar, onde é aplicada aos valores do histograma de cor no espaço de cor HSV (referido na subsecção 2.3.3.1). De uma forma mais especifica, o descritor de cor escalável, extrai, normaliza e mapeia de forma não linear os valores do histograma, numa representação inteira a 4-bit, dando assim mais relevância a valores mais pequenos. A transformada de Haar, é assim aplicada aos valores inteiros a 4-bit através das barras do histograma.

A extração do descritor é realizada com computação um histograma de cor com 256 níveis no espaço de cor de HSV com a componente matiz (H) quantificada a 16 níveis, e a saturação (S) e o valor (V) quantificado cada um para 4 níveis [19].

A aplicação típica do descritor, inclui como por exemplo, a busca de similaridade numa base de dados com conteúdo multimédia e pesquisa em enormes base de dados.

2.3.3.4 Grupo de *Frames* / Grupo de Imagens

O descritor de cor Grupo de Frames / Grupo de Imagens (GoF/GOP) é sobretudo utilizado para a representação conjunta de cores para várias imagens ou várias *frames* de um segmento de vídeo, contíguas ou não contíguas. Este baseia-se em histogramas, que capturam de forma confiável o conteúdo da cor de várias imagens ou *frames* de vídeo. Normalmente para um grupo de *frames* ou imagens, é selecionado uma frame chave ou imagem chave, que representará as características relacionadas com o grupo. Os métodos são altamente dependentes da qualidade da

seleção da amostra representativa, o que pode levar a resultados pouco fiáveis caso não seja bem executada [21].

A estrutura do descritor GoF / GoP é idêntica à do cor escalável, com a exceção do campo agregação, que especifica como os pixels da cor de diferentes imagens/*frames* foram combinadas antes da extração do histograma de cor. Os valores possíveis são média, mediana e cruzamento.

Uma das aplicações deste descritor, é a pesquisa em conjuntos grandes de imagens, para encontrar *clusters* de imagens semelhantes através da cor, em que é utilizado a interseção de histogramas como medida de similaridade da cor. A interseção de histogramas é obtido calculando o valor mínimo de cada barra de cor do histograma ao longo das *frames*/imagens e atribui esse valor às barras de cor do histograma resultante. A interseção encontra as cores mínimas comuns nas *frames*/imagens, e portanto, pode ser utilizado em aplicações que requerem a deteção de um elevado grau de correlação da cor [19].

2.3.3.5 Estrutura de cor

Este descritor é uma generalização do histograma de cores, que apresenta algumas características espaciais da distribuição de cores em uma imagem. Este tem a particularidade de, para além de apresentar o conteúdo da cor de forma semelhante a um histograma de cor, também apresentar informações sobre a estrutura de uma imagem, sendo esta a característica diferenciadora deste descritor de cor. Em vez de considerar cada pixel separadamente, o descritor recorre a uma estrutura de 8x8 pexels que desliza sobre a imagem. Ao contrário do histograma de cor, este descritor consegue distinguir duas imagens em que uma determinada cor está presente em quantidades iguais, mas que apresenta uma estrutura num dos grupos de pixels 8x8 com uma cor diferente nas duas imagens. Os valores de cores são representadas no espaço de cor HMMD com cone duplo, sendo o espaço quantificado de maneira não uniforme em 32, 64, 128 ou 256 níveis. Cada valor de amplitude de um nível é representado por um código de 8 bits. Este descritor apresenta um bom desempenho na tarefa de recuperação de imagens baseado na similaridade [22].

2.3.3.6 Disposição de cor

O descritor de disposição de cor, caracteriza a distribuição espacial de cor de uma imagem. Este usa um vetor de cores representativas de uma imagem, expressas no espaço de cor YCbCr. O tamanho do vetor é fixado em 8x8 elementos para garantir invariância da escala do descritor. As cores representativas da imagem podem ser selecionadas de diversas maneiras, mas a mais simples é através do cálculo da média de cor do bloco de imagem correspondente. O descritor disposição de cor, pode ser usado para em pesquisa rápida de bases de dados de imagens [20].

2.3.4 Descritores de textura

A textura das imagens é uma característica visual importante, que tem muitas aplicações na recuperação, navegação e indexação de imagens. Existem três descritores de textura, referenciados na norma MPEG-7 [2]. Em seguida é apresentada uma pequena descrição dos mesmos.

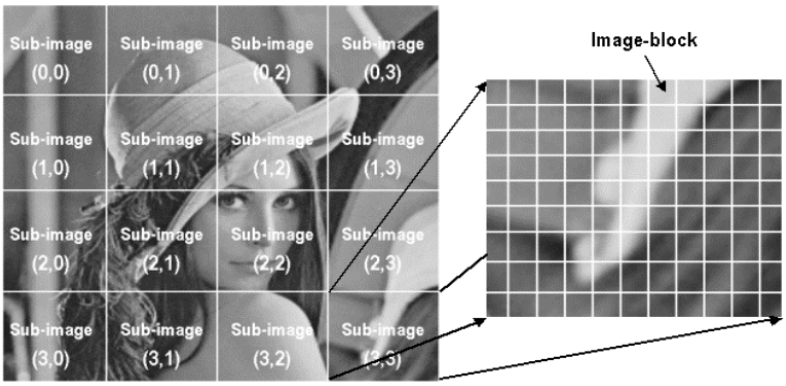


Figura 2.4: Sub-imagem e bloco de imagem. *Retirada de* [2]

2.3.4.1 Descritor de textura homogénea

O descritor de textura homogénea (HTD - *Homogeneous Texture Descriptor*) descreve a distribuição estatística da textura de uma imagem. Existem neste descritor 62 interfaces de recurso, sendo 2 no domínio espacial e 60 no domínio das frequências. No domínio espacial, é extraído a média e o desvio padrão de uma imagem. No domínio das frequências, o espaço é dividido em 30 canais, sendo calculado o valor energético e o valor do desvio de energia da resposta do filtro Gabor em cada canal [2, 23]. Este desenho baseia-se no facto de a resposta do córtex visual possuir banda limitada e do facto de o cérebro decompor o espectro em bandas na frequência espacial [2]. O descritor de textura homogénea é essencialmente utilizado em aplicações de recuperação de imagens por similaridade.

2.3.4.2 Descritor de histograma de borda

O descritor de histograma de borda (EHD - *Edge Histogram Descriptor*) apresenta-se sobre a forma de um histograma de 80 níveis, que representa a distribuição de borda local de uma imagem. Este descreve as bordas em cada sub-imagem. Estas sub-imagens são obtidas através da divisão da imagem numa grelha 4x4 como pode ser visto na figura 2.4. Existem 5 tipos de classificação diferentes das bordas de cada sub-imagem, sendo elas: vertical, horizontal, 45-graus, 135-graus e não direcional [2].

Este descritor é utilizado na recuperação de imagens, como por exemplo, imagens naturais ou de esboço, devido à sua textura homogénea. É também suportado por este descritor, a pesquisa baseada em blocos de imagem.

2.3.4.3 Descritor de navegação percetual

O descritor de navegação percetual (HTD - *Perceptual Browsing Descriptor*) foi projetado para navegação em base de dados, mas principalmente para quando essa navegação necessita de recursos com sentido percetual [2]. Este descritor é bastante compacto, que requer apenas 12 bits (máximo) para caracterizar regularidade (2 bits), direcionamento (3 bits x 2) e grosseirismo

(2 bits x 2) da textura de uma imagem. A regularidade de uma textura pode apresentar valores numa escala entre 0 e 3, em que 0 indica uma textura irregular ou aleatória e 3 indica um padrão com direção e grosseirismo bem definidos. O direcionamento de uma textura é quantizada em 6 valores, variando de 0 a 150 em degraus de 30. Por fim, o grosseirismo de uma textura está relacionado com a escala e resolução de uma imagem. É quantizado em 4 níveis de 0 a 3, sendo 0 para um grão fino e 3 para uma textura grosseira. Estes valores têm uma relação com a divisão do espaço de frequência usado no cálculo do descritor de textura homogénea (HTD) [18]

2.3.5 Descritores de forma

O descritores de forma são dos descritores mais poderosos no reconhecimento de objetos. Isto deve-se ao facto de os seres humanos serem exímios no reconhecimento de objetos característicos exclusivamente através da suas formas, provando que a forma muitas vezes possui informação semântica [3].

2.3.5.1 Descritor de forma baseado em região

O descritor de forma baseado em região (RSD - *Region-based Shape Descriptor*) [3] apresenta a distribuição de um pixel dentro de uma região de um objeto em 2 dimensões. Este permite a descrição de objetos simples, com ou sem buracos (figura 2.5), mas também permite a descrição de objetos mais complexos, que contém múltiplas regiões sem ligação.

Figura 2.5: Exemplo de formas de objetos que podem ser descritas eficazmente pelo descritor baseado em região. *Retirada de* [3] .

As principais características deste descritor são [3]:

- Fornece uma forma compacta e eficiente de descrever várias regiões disjuntas;
- Quando, no processo de segmentação de um objeto, ocorrem sub-regiões sem ligação, o objeto ainda pode ser recuperado, desde que a informação de quais as regiões que foram divididas seja mantida e usada na extração do descritor;
- Apresenta uma boa robustez à segmentação de ruído.

2.3.5.2 Descritor de forma baseado no contorno

O descritor de forma baseado no contorno (CSD - *Contour-based Shape Descriptor*) [3] fundamenta-se na representação da curvatura espaço-escala (CSS - *Curvature Scale-Space*) do contorno. O contorno é uma propriedade importante na identificação de objetos semanticamente semelhantes. É também bastante eficiente em aplicações onde a forma de objetos são muito variáveis, ou quando por exemplo, existem deformações de perspetiva. Esta apresenta um boa eficiência mesmo perante a existência de ruído nos contornos.

As principais características deste descritor são [3]:

- Consegue distinguir objetos que apresentem formas semelhantes mas que a forma do contorno apresenta propriedades bem diferenciadoras;
- Tem a capacidade de encontrar formas que são semanticamente similar para os seres humanos, mesmo quando existe uma significativa variabilidade intra-classe;
- É eficiente mesmo em casos de deformações não rígidas;
- É eficiente mesmo em casos de distorções do contorno devido a variações de perspetiva, sendo uma situação muito comum em imagens e vídeo.

2.3.6 Descritores locais

Um descritor local permite a localização das estruturas locais de uma imagem de forma repetitiva. Estas são codificadas de modo a que sejam invariantes a transformações das imagens, tais como a translação, rotação, mudanças de escala ou deformações. Assim, estes descritores podem ser utilizados para representar uma imagem e podem ser utilizados para diversos fins, tais como, reconhecimento de objetos, reconhecimento de cenas, perseguição de movimento, correspondência entre imagens ou mesmo obtenção de estruturas 3D de múltiplas imagens. Para a extração das características deste descritor é necessário utilizar um processo com as seguintes etapas [24]:

- Encontrar um conjunto de pontos chave;
- Definir uma região em torno de cada ponto-chave numa escala invariante;
- Extrair e caracterizar o conteúdo da região;
- Calcular o descritor da região normalizada;
- Combinar os descritores locais.

Em seguida são apresentados duas das técnicas mais representativos dos descritores locais, o SIFT e o SURF.

2.3.6.1 SIFT

O descritor SIFT (*Scale-invariant feature transform*) [25, 4] é um descritor local que transforma uma imagem numa grande coleção de vetores de características locais invariantes a translação, rotação, mudanças de escala, e parcialmente invariante a mudanças de iluminação. Pode assim ser utilizado para detetar correspondência entre imagens com diferentes visões de objetos ou cenas. Este descritor apresenta como característica interessante o facto de compartilhar uma série de propriedades em comum com as respostas dos neurónios do lobo temporal na visão dos primatas. Os pontos-chave SIFT derivados de uma imagem são usados na indexação numa abordagem de vizinho mais próximo, para encontrar objetos candidatos. O modelo recorre a uma votação pela transformada Hough e a uma estimativa final pelo método dos mínimos quadrados. Quando 3 pontos chave estiverem em acordo, pode se afirmar que existe uma forte possibilidade da presença de um objeto. Como foi indicado anteriormente, são necessários levar a cargo uma lista de etapas bem definidas para obtenção de descritores locais, sendo que as etapas para o descritor SIFT as seguintes:

1. **Deteção de extremos**: A deteção de extremos (máximos e mínimos) é conseguida através da procura realizada em várias escalas e localizações, de modo a serem extraídos pontos de interesse invariáveis à escala e rotação, através da diferença de filtros gaussianos. Estes pontos de interesse ou pontos chave correspondem a estes extremos para várias escalas. Um filtro gaussiano passa baixo é dado pela convolução entre uma imagem I e a função G:

$$L(x,y,\sigma) = G(x,y,\sigma) * I(x,y) \tag{2.17}$$

onde ,

$$G(x,y,\sigma) = \frac{1}{2\pi\sigma^2} e^{\frac{-\left(x^2+y^2\right)}{2\sigma^2}} \tag{2.18}$$

em que o o filtro varia à escala através do parâmetro teta

A função *DoG* (*"Diference of Gaussian"*) é dada pela diferença entre as imagens filtradas em escalas próximas separadas por uma constante k e pode ser definida como:

$$DoG(x,y,\sigma) = G(x,y,k\sigma) * G(x,y,\sigma) \tag{2.19}$$

Assim, a convolução de uma imagem I com o filtro *DoG* é dado por:

$$D(x,y,\sigma) = (G(x,y,k\sigma) - G(x,y,\sigma)) * I(x,y) = L(x,y,k\sigma) - L(x,y,\sigma) \tag{2.20}$$

que corresponde à diferença entre as imagens filtradas pelo filtro gaussiano em escalas teta e $k\sigma$. Este filtro provoca uma perda de nitidez nas imagens (efeito desfoque) e tornasse assim capaz de detetar variações de intensidade nas imagens, como por exemplo, nos contornos.

A figura 2.6 mostra como é realizado o processo de obtenção das Diferenças Gaussianas e consequente formação das oitavas.

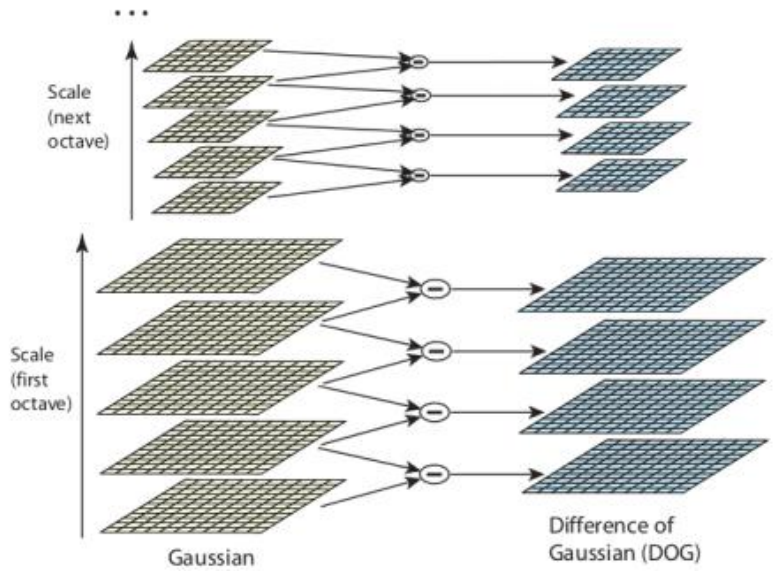


Figura 2.6: Construção das Diferenças Gaussianas e formação de oitavas *Retirada de* [4] .

Segundo Lowe [4] é necessário atingir a escala 2σ para ser possível a construção de um descritor local invariável à escala, logo

$k = 2^{(1/s)}$

onde s é o número de intervalos entre imagens obtidas por *DoG* e $D(x,y,\sigma)$ corresponde à primeira imagem e $D(x,y,2\sigma)$ à última de todo conjunto de imagens geradas. Também deverão ser assim obtidas $s+3$ imagens na pilha de imagens filtradas para cada oitava. Cada oitava contém as imagens de Diferença Gaussiana, sendo as que ficam entre as escalas superiores e inferiores designadas de intervalo.

Por fim é realizado o processo de deteção de extremos, onde um pixel é comprado com os seus oito vizinhos na imagem atual e com os nove pixeis vizinhos das imagens de escalas adjacentes, numa região de 3x3. A figura 2.7 ilustra este processo.

O próximo processo passa pela localização dos ponto chave.

2. **Localização de pontos chave**: quando detetado um extremo, esse ponto é considerado um candidato a ponto chave ou ponto de interesse. Este ponto foi encontrado através da comparação de um pixel com os seus vizinhos como referido anteriormente. sendo necessário realizar um cálculo de ajuste detalhado da localização e escala gaussiana de cada um destes pontos. É utilizada então a série de Taylor para obter uma localização mais exata dos extremos, sendo rejeitado caso a intensidade de um extremo seja inferior a um limiar previamente definido.

   Como DoG tem uma boa resposta a arestas e como estás fazem com que os pontos sejam instáveis com ruído, estes necessitam de ser removidos. É assim através da utilização uma

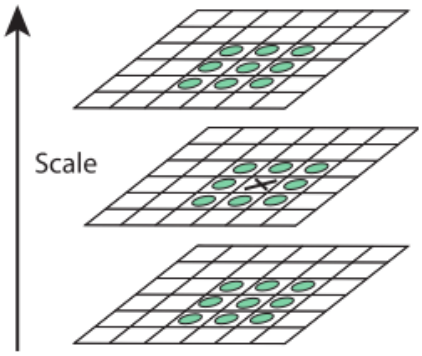


Figura 2.7: Ilustração do processo de deteção de máximos e mínimos das imagens de Diferença Gaussiana. O pixel candidato está marcado com X e os vizinhos com um circulo. *Retirada de* [4]

matriz Hessiana 2x2 é possível com calcular as curvaturas principais. Caso o rácio seja superior a um limiar previamente definido, é considerado uma aresta e assim o ponto chave é descartado.

Após a remoção de todos os pontos considerados não sendo de interesse, fica-se com todos os pontos chave, sendo necessário a atribuição das suas orientações.

3. **Atribuição da orientação dos descritores**: este processo tem como principal finalidade possibilitar a representação de um descritor em relação a sua orientação, permitindo assim que este seja invariante a rotações. Para realizar esta tarefa é utilizada a escala Gaussiana σ para a escolha da imagem filtrada L com a escala mais próxima e com a oitava referente ao ponto avaliado, tornando assim invariante também à escala.

   Assim são calculados os gradientes para cada imagem $L(x,y,\sigma)$ de intervalo, referentes às escalas e oitavas utilizadas.

   A magnitude e orientação são calculados da seguinte forma:

$$m(x,y) = \sqrt{\Big((L(x+1,y) - L(x-1,y))^2 + (L(x,y+1) - L(x,y-1))^2\Big)} \tag{2.21}$$

$$\theta(x,y) = tan^{-1}\left(\frac{(L(x,y+1) - L(x,y-1))}{(L(x+1,y) - L(x-1,y))}\right) \tag{2.22}$$

   Assim, é criado um histograma das orientações para pixeis numa região em redor do pontos chave, em que de todas as orientações obtidas para um ponto, apenas o maior pico e aquelas acima de 80% do valor desse pico é que são utilizadas para definir a orientação de cada ponto chave.

   Por fim, é possível a construção dos descritores para os pontos chave definidos como o ponto a seguir apresenta.

Figura 2.8: Imagem

4. **Construção do descritor local**: este é o ultimo passo em que é criado o descritor local para cada ponto de interesse. Para esse processo, é considerado um bloco 16x16 em redor do ponto chave e posteriormente dividido em 16 sub-blocos de 4x4. Por cada sub-bloco é criado um histograma com 8 picos relativos à orientação. Isto faz que no fim seja extraído um vetor de 128 posições para cada ponto chave. Para além deste retorno, são também tomadas várias medidas de modo a que exista robustez suficiente para que esse ponto de interesse seja invariante a mudanças de iluminação e rotação.

O algoritmo SIFT é regularmente utilizado em reconhecimento de objetos ou cenas, tendo sido utilizado em deteção de objetos em *frames* de vídeo [26, 27] com utilização de técnicas mais avançadas de deteção de objetos e cenas utilizando vocabulários visuais como é apresentado na secção 2.3.7.

2.3.6.2 SURF

Outro algoritmo importante de extração de descritores locais é o SURF (*Speeded Up Robust Features*). Este é baseado no SIFT apresentado na subsecção anterior 2.3.6.1 e também é utilizado, por exemplo, em reconhecimento de objetos ou mesmo na reconstrução 3D. Segundo os autores [5] o SURF é mais rápido (cerca de dez vezes) e robusto do que o SIFT, sendo que, segundo a comparação realizada em [28] comprovou-se que o SIFT é mais lento e não muito bom a mudanças de iluminação, mas apresenta melhores resultados a variações de rotação, mudanças de escala e transformações na imagem.

Este usa a técnica da imagem integral, onde cada pixel de uma imagem recebe um valor igual à soma dos pixeis da sua esquerda e acima, incluindo o próprio. Este também utiliza um filtro Haar em formato de caixa numa sub-região 4x4 em redor de um ponto de interesse, como se pode ver na figura 2.9, tornando o processo computacionalmente eficiente. Isto é realizado calculando a soma das respostas dos filtros e a soma do modulo das respostas dos filtros nas direções horizontal e vertical, gerando assim 4 valores por cada sub-região. Logo, o SURF retorna um vetor de 64 dimensões, metade do retornado pelo algoritmo SIFT.

Para além destes dois algoritmos ainda existe outros menos utilizados como o GLOH (*Gradient Location and Orientation Histogram*) e o HOG (*Histogram of Oriented Gradients*).

Na próxima secção e apresentado

2.3.7 Descritores baseado em vocabulário visual

No reconhecimento de documentos de texto é utilizado o conceito de vocabulário textual, sendo muitas vezes designado por *bag of words* que em português significa, saco de palavras, onde existe um conjunto de palavras armazenadas pré definidas. Um texto é assim caracterizado, através da análise das palavras que possui, sendo contabilizado a frequência de palavras que estejam

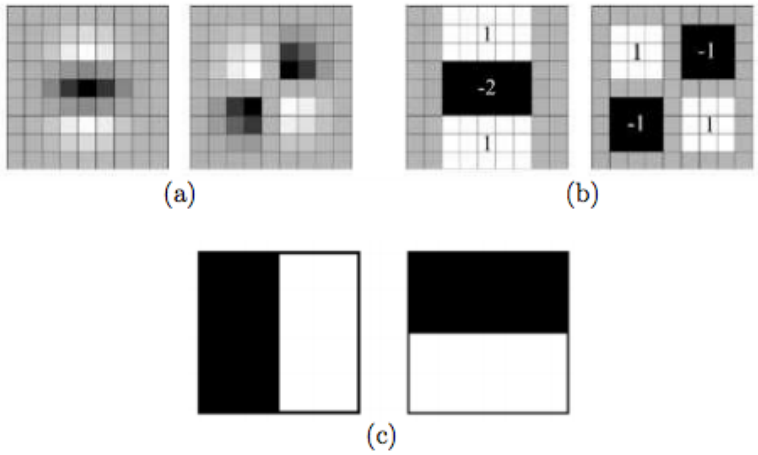
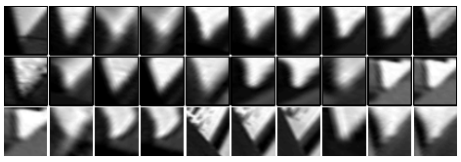


Figura 2.9: (a) Filtros Gaussianos de segunda ordem nas direções yy e xy; (b) aproximação por filtros de caixa; (c) filtros de Haar; As regiões a cinzento têm valor igual a zero. *Retirada de* [5].

simultaneamente no texto e no *bag of words*, que assim atribuem um significado ao texto. É também utilizado uma lista de palavras que não acrescentam significado a expressões tais como, "o"ou "um", entre outras, que são removidas do texto durante a análise para não influenciarem os resultados. Um sistema de extração de informação de texto apresenta um número padrão de etapas [29].

Recentemente esta técnica foi adotada em aplicações de extração de informação visual, como por exemplo é nos mostrado em [26, 27]. Esta recorre a descritores locais invariantes a escala e rotação, para criar um vocabulário visual. A utilização de descritores locais como o SIFT, referido na secção anterior anterior 2.3.6.1, permite a extração de pontos de interesse nas imagens, invariantes a rotação e mudanças de escalas.

Quando efetuado este processo repetidamente com um grande conjunto de imagens, é possível criar *clusters* de regiões de imagens muito semelhantes entre si, como se pode ver na figura **??**. Todas estas regiões são candidatas a palavras visuais, sendo selecionada a que representa melhor esse *cluster*, isto é, é selecionado o centroide desse conjunto de regiões semelhantes entre si, recorrendo ao algoritmo *k-means* referido na secção 2.1.1. Esse centroide passa então a ser considerado uma palavra visual e é adicionado a um vocabulário com outras palavras visuais.



Por fim é necessária a realização de uma indexação de cada palavra visual às imagens, sendo utilizado um vetor para cada imagem que indica, o número de vezes em que uma determinada palavra visual se repete numa imagem. Neste caso o vetor funciona como em *text mining*, em que existe a contagem da frequência de palavras que ocorrem num determinado documento.

Outro processo possível para indexação do conteúdo visual ou texto, é atribuição de um peso ou ponderação para cada palavra visual numa determinada imagem. Aqui é utilizado a ponderação padrão conhecida como tf-idf ('*term frequency-inverse document frequecy*') [26].

Considerando um vetor com k palavras visuais $Vd = (t_1, ..., t_i, ..., t_k)$, em que a ponderação de cada palavra é dada pela equação 2.23

$$t_i = \frac{n_{id}}{n_d} log \frac{N}{n_i} \tag{2.23}$$

onde n_{id} é o numero de ocorrências da palavra i num documento d, n_d o número total de palavras no documento d, n_i é o numero de ocorrências da palavra i em todos documentos e N é o numero de documentos existentes.

Concluí-se assim que este descritor permite a identificação de imagens semelhante ou reconhecimento de objetos, através da comparação dos vetores de cada imagem, com a informação relativa ao vocabulário posteriormente criado. Este demonstra ser bastante eficiente, e segundo Nistér e Stewénius [30] é possível escalar este processo para enormes quantidades de imagens sem perda de performance utilizando para isso um vocabulário em forma de árvore, isto é, com a hierarquização das palavras visuais de um vocabulário visual, recorrendo para isso ao *clustering* hierárquico.

Capítulo 3

Módulo da informação visual

Neste capítulo será introduzido o módulo da informação visual desenvolvido, sendo apresentado a estrutura e organização deste sistema. Como referido no capítulo 1, este projeto pretende estender a ferramenta TweeProfiles descrita na secção 2.2, dando-lhe uma dimensão de conteúdo diferente à que possuí, em que a informação tido em conta, em conjunto com a espacial e temporal, passa a ser imagens partilhadas em *tweets* e não o texto. Para que isto seja realizável, foi necessário o desenvolvimento de um módulo que efetue a recolha os dados com a devida filtragem, extraia e processe a informação visual e armazene essa informação de modo a que fosse possível a sua integração com o TweeProfiles.

3.1 Recolha dos dados

O desenvolvimento deste módulo apenas era realizável com um conjunto de imagens partilhadas no serviço de microblogging Twitter, sendo fundamental a recolha dos dados necessários para esse processo, neste caso, os Tweets. Esta secção é feita uma descrição do conjunto dos dados disponíveis, das filtragens que foram necessárias realizar e uma apresentação do conjunto de dados finais obtidos e utilizados no desenvolvimento deste projeto de dissertação.

3.1.1 Descrição dos dados

O primeiro passo para a realização deste projeto de dissertação foi a recolha dos dados necessários. Estes dados foram recolhidos através de uma base de dados mongodb previamente criada usando a plataforma Socialbus, anteriormente designada por TwitterEcho [10]. Este dados vêm sobre a forma de objetos JSON e possuem informação relativa a cada tweet como pode ser visto no anexo A. Estes objetos encontram-se todos num só documento mongodb que se caracteriza pelas características apresentadas na tabela 3.1

	Total	Com imagem	Twitter	TwitPic	Instagram
Nº Tweets	1704273	86349	202	6100	79210

Tabela 3.1: Descrição em números do total de tweets com indicação, nos que contém URL para imagem, do número de tweets por serviço de partilha de imagem

Estes tweets foram recolhidos entre o dia 17 e 19 de Junho de 2013 com conteúdo partilhado somente contendo texto escrito em português do Brasil, tendo sido isto foi possível graças á capacidade de filtragem da ferramenta Socialbus [10] já referida anteriormente. Estas datas coincidiram com um evento ocorrido no Brasil, mais especificamente, as manifestações do ano passado do povo brasileiro contra o seu governo. Este foi um dos motivos da escolha desta base de dados, pois apresentava tweets que poderiam ser interessante para encontrar padrões ou eventos através das imagens partilhadas pelos brasileiros nas ruas, aliadas sempre às dimensões espaço-temporais.

3.1.2 Filtragem dos dados

Após estar definido o conjunto de dados a utilizar, foi necessário realizar uma filtragem dos dados de modo a apresentem a informação necessária para a realização deste projeto. O primeiro passo desta filtragem foi recolher todos os tweets que contivessem no seu objeto um URL para uma imagem, sendo que esse URl teria de pertencer a um dos seguintes serviços:

- Twitter
- TwitPic
- Instagram

e teria que esse URl ser válio, isto é, foi feita uma prévia verificação se a imagem estaria ainda disponível através do endereço existente.

Para ser mais fácil posteriormente uma seleção mais cuidadosa dos tweets foi criada uma base de dados local (SQLite) com a seguinte tabela:

```
create table if not exists IMAGENS (
  id integer PRIMARY KEY AUTOINCREMENT,
  id_tweet text,
  servico text,
  url text,
  tipo text,
  retweet text
);
```

Em que se descreve cada coluna da seguinte forma:

id - id da linha da tabela;

id_tweet - id do tweet na base de dados Mongodb;

servico - nome do serviço de alojamento da imagem;

url - endereço url para a imagem fonte;

tipo - este atributo identifica se a imagem pertence a um tweet ou retweet;

retweet - caso a imagem pertença a um retweet, este atributo pode assumir o valor "primeiro"no caso de ser o primeiro retweet, do tweet original, na base de dados mongodb, ou caso contrário, assume o valor NULL.

Isto permitiu realizar de uma forma rápida alguns teste no download de algumas imagens pelos diferentes serviços, para além de permitir fazer uma seleção fácil e rápida de tweets, retweets ou por exemplo, do primeiro retweet no caso de existir vários retweets de um determinado tweet.

Após a criação desta base de dados, ficou decidido selecionar todos os objetos da base de dados Mongodb do tipo tweet em que o seu serviço fosse o Instagram. Esta decisão deveu-se ao facto da inclusão de retweets apenas introduzir imagens já existente ocorrendo apenas duplicação de dados, no caso da escolha do Instagram, deveu-se ao facto de este apresentar um número superior de imagens relativamente aos outros serviços, visto que ao se excluir os retweets o número de imagens dos restantes serviços passou a um valor pouco significativo.

Para além destes critérios é importante salientar que foi necessário selecionar apenas os dados com georreferenciação.

3.1.3 Conjunto de dados final

Por fim armazenou-se um ficheiro JSON com todos os dados a ser utilizados e foi realizado o download de todas as imagens relativas a cada tweet e armazenadas localmente em formato JPEG, que se apresentava como formato de origem das imagens descarregadas.

Relativamente aos objetos de cada tweet presentes no ficheiro JSON, optou-se por não armazenar todas as instâncias para reduzir o tamanho do ficheiro, tendo sido apenas incluídas as representadas no seguinte exemplo de um objeto de um tweet:

```
{
    "_id": {
        "$oid": "52c6d0f08ef20d397e42b516"
    },
    "coordinates": {
        "coordinates": [
            -8.61136747,
            41.14668427
        ]
    },
    "created_at": "Tue Jun 18 17:02:09 +0000 2013",
    "entities": {
        "urls": [
            {
                "display_url": "instagram.com/p/atTgTbEu0S/"
```

```
            }
        ]
    },
    "id_str": "347036525172772864",
    "text": "#ogiganteacordou #DilmaNAO #brasilnarua #foradilma #
    verasqueumfilhoteunaofogealuta #vamospararuas\u2026 http://t.co/AF0hcTxUBX"
    ,
    "user": {
        "id_str": "1072354428",
        "name": "Carlos Roma"
    }
}
```


No caso das imagens armazenadas, de modo a que fosse associada cada uma das imagens ao seu respetivo tweet, foi atribuído o campo *id_str* presente no objeto JSON ao nome do ficheiro de imagem JPEG. Assim ao utilizar uma das imagens, para saber a que tweet pertence apenas é necessário procurar o objeto JSON que possua o atributo *id_str* igual ao nome do ficheiro da imagem.

Recolhido todos os identificadores dos tweets pertencentes ao serviço Instagram que apenas fossem do tipo tweet e que os respetivos tweets apresentassem a informação de geolocalização no campo *coordinates*, avançou-se com o processo de *download* de todas as imagens, tendo sido efetuado com sucesso o download de 5958 imagens em 7195 possíveis.

Em suma, o conjunto de dados final utilizado para o desenvolvimento deste projeto são 5958 objetos relativos aos tweets contidos num ficheiro JSON e as respetivas 5958 imagens associadas a cada tweet.

3.2 Extração, processamento e armazenamento da informação visual

O passo seguinte no desenvolvimento do módulo da informação visual foi a implementação de um sistema capaz de extrair, processar e armazenar a informação visual através das imagens armazenadas localmente. Para o desenvolvimento deste modulo foi utilizada a linguagem Python, pela sua capacidade de integração de diferentes bibliotecas desde manipulação de estruturas de dados, até mesmo a bibliotecas de manipulação e processamento de imagem. Para o a concretização modelo foi tido como linhas guia, o processo desenvolvimento de uma ferramenta de pesquisa de imagens apresentado em [31]. A arquitetura deste módulo desenvolvido é apresentada na figura 3.1.

Como se pode ver pela figura 3.1 foram desenvolvidos três sub-módulos diferentes. O primeiro é o responsável pela extração dos descritores locais, o segundo utiliza o primeiro da gerar um vocabulário visual, e o terceiro e último sub-módulo utiliza os dois anteriores para armazenar um histograma descritor de cada imagem numa base de dados indexada a sua respetiva imagem. Em seguida é apresentada uma subsecção com a descrição para cada sub-módulo, como representado na figura 3.1

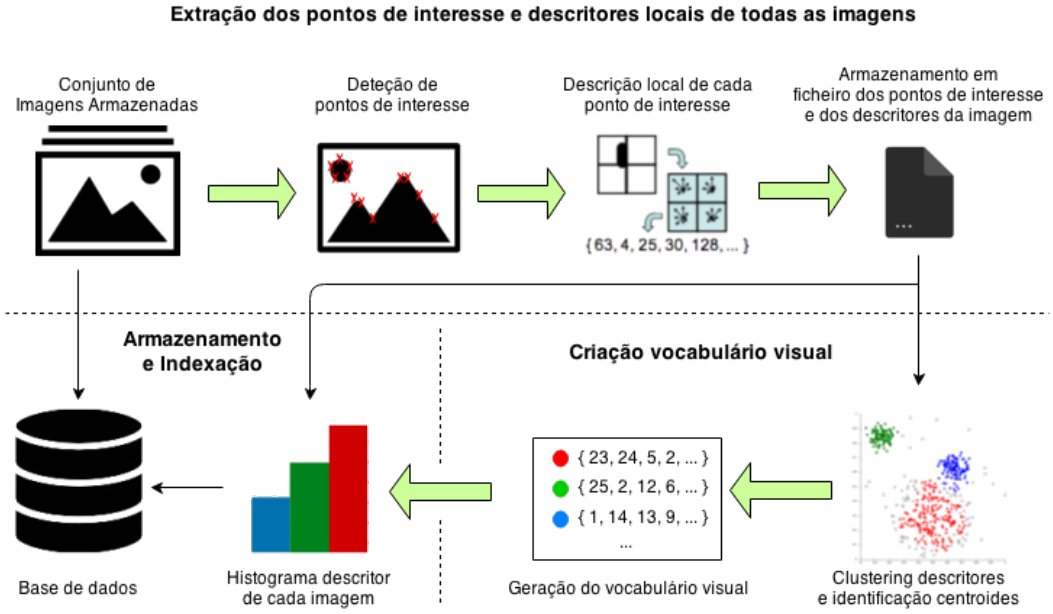


Figura 3.1: Arquitetura do módulo de extração, processamento e armazenamento da informação visual. *Adaptada de* [6]

3.2.1 Extração dos pontos de interesse e descritores locais

O primeiro passo no desenvolvimento deste módulo passou pela extração da informação visual. Esta informação visual deveria representar uma imagem eficientemente e de forma a possibilitar a criação de um vocabulário visual. Uma das formas possíveis e apresentadas na secção 2.3 é a utilização de um descritor local. Neste caso foi utilizado o descritor SIFT [25, 4], pois como referido na subsecção 2.3.6.2, apesar de ser mais lento e menos eficiente a mudanças de iluminação, este apresenta melhores resultados a variações de rotação, mudanças de escala e transformações na imagem. Para além disso, foi utilizada a ferramenta e biblioteca *open source* VLFeat [32] que integra alguns dos algoritmos mais utilizados em visão computacional, e que inclui o algoritmo SIFT. Este, apesar de não possuir uma biblioteca para Python, permite a sua utilização através da linha de comandos.

Para o conjunto de imagens existentes foi necessário criar uma copia de cada imagem em escalas de cinzento e em formato *.pgm* para ser utilizada pelo VLFeat. Utilizando assim estas imagens, o VLFeat armazena num ficheiro com o formato *.sift* os pontos de interesse e os descritores de uma imagem, sendo necessário criar um ficheiro para cada imagem. Nesses ficheiros os dados são armazenados em formato ASCII. A informação armazenada apresenta-se da seguinte forma

```
318.861 7.48227 1.12001 1.68523 0 0 0 1 0 0 0 0 0 11 16 0 ...
318.861 7.48227 1.12001 2.99965 11 2 0 0 1 0 0 0 173 67 0 0 ...
54.2821 14.8586 0.895827 4.29821 60 46 0 0 0 0 0 0 99 42 0 0 ...
155.714 23.0575 1.10741 1.54095 6 0 0 0 150 11 0 0 150 18 2 1 ...
42.9729 24.2012 0.969313 4.68892 90 29 0 0 0 1 2 10 79 45 5 11 ...
```

```
229.037 23.7603 0.921754 1.48754 3 0 0 0 141 31 0 0 141 45 0 0 ...
232.362 24.0091 1.0578 1.65089 11 1 0 16 134 0 0 0 106 21 16 33 ...
201.256 25.5857 1.04879 2.01664 10 4 1 8 14 2 1 9 88 13 0 0 ...
... ...
```

onde cada linha contêm as coordenadas, escala e ângulo de rotação para cada ponto de interesse, nos primeiros 4 valores respetivamente, correspondendo os restantes ao vetor descritor de tamanho 128 como referido no capítulo 2 na subsecção 2.3.6.1. Como o objetivo era utilizar as imagens originais, foram eliminadas as imagens temporárias com o formato *.pgm*. O passo seguinte passou pelo desenvolvimento do submodelo responsável pela criação do vocabulário visual, que é apresentado na subsecção seguinte.

3.2.2 Criação do vocabulário visual

Este módulo é o responsável pela criação do vocabulário visual. Para a sua concretização foi necessário utilizar os ficheiros de extensão *.sift* reproduzidos através da ferramenta VLFeat [32] no módulo anterior. Os passos seguintes basearam-se nos descritos em [31].

As palavras visuais não são nada mais do que um conjunto de vetores de características de imagens. Assim um vocabulário visual é o conjunto destas palavras visuais. Como todas as imagem possui muitos descritores locais, sendo que muitos podem ser semelhantes, é necessário agrupar todos os descritores de um conjunto de imagens e detetar aqueles que possam representar um conjunto de descritores semelhantes, e assim formar várias palavra visual, sendo uma palavra visual um centroide de um grupo.

Para criar um vocabulário visual foi então necessário utilizar um algoritmo de *clustering* por partição, tendo sido escolhido o *k-means* por ser um dos mais utilizados e eficiente, como foi referido no capítulo 2 na secção 2.1.1. O algoritmo foi aplicado aos descritores de um subconjunto de imagens aleatoriamente selecionadas do conjunto de imagens armazenadas localmente. Neste caso foi utilizado aproximadamente 8% das imagens para não comprometer a nível de tempo de processamento. Como este algoritmo implica a pré definição do número de *clusters*, foi atribuído a k o valor 1000. Isto significa que são gerados cerca de 1000 *clusters*, logo serão retornados aproximadamente 1000 centroides, o que significa que o nosso vocabulário visual possuirá cerca de 1000 palavras visuais.

Para utilizar este vocabulário visual é necessário armazena-lo e indexar cada palavra visual a cada imagem. O sub-módulo responsável por este processo é descrito na secção a seguir.

3.2.3 Armazenamento da informação visual

Com o vocabulário visual criado foi necessário armazenar esta informação e indexar cada palavra visual as imagens, de modo a seja possível a comparação entre imagens diferentes. Para armazenar este vocabulário desenvolveu-se módulo que execute esta tarefa, sendo que, foi necessário seguir alguns passos tais como, a criação de um histograma descritor do vocabulário visual

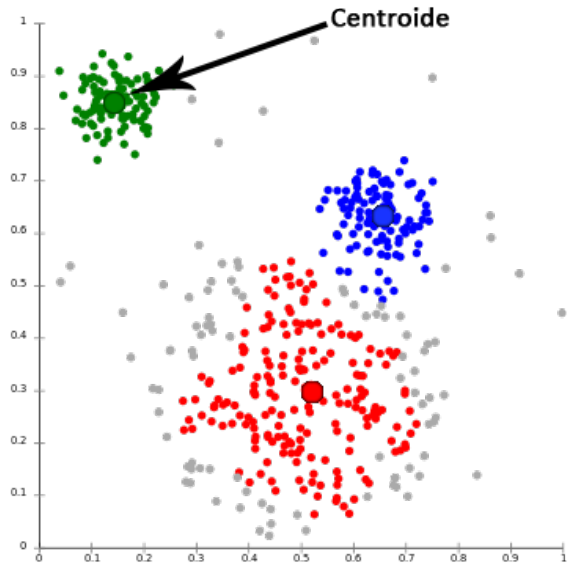


Figura 3.2: Exemplo de projeção de centroides após tarefa de *clustering* num espaço a duas dimensões.

de cada imagem e criação de uma base de dados com a informação necessária para a utilização do histograma descritor de cada imagem.

A criação do histograma foi o primeiro passo do desenvolvimento deste sub-módulo, em que foi utilizado o vocabulário disponibilizado através do sub-módulo apresentado anteriormente. Como o vocabulário é constituído por vetores descritores que representam cada uma das palavra visual, para a criação do histograma para cada imagem foi necessário utilizar novamente os ficheiros com os descritores locais SIFT de cada imagem e assim projetar num espaço utilizando novamente o algoritmo *k-means* de modo a atribuir a cada ponto chave de uma imagem a sua palavra visual respetiva. Foi então possível após este processo criar um histograma com a contagem das palavras visuais em cada imagem.

Assim o próximo passo passou pelo armazenamento dos histograma e a indexação a sua respetiva imagem. Como todo este módulo foi operado em modo *offline*, optou-se pela utilização de uma base de dados local SQLite. Esta funciona de modo semelhante a uma base de dados MySQL ou PostgreSQL, sendo que pode ser desenvolvida e acedida sem recurso a um servidor. Para a sua concretização foi criado um esquema muito simples apenas com três tabelas como ilustrado na tabela 3.2. A tabela *imlist* contém o nome de todas as imagens através do atributo *filename*, a tabela *imwords* contém índice das palavras visuais através do atributo *wordid*, a identificação do vocabulário com o atributo *vocname* e o índice das imagens em que as palavras visuais aparecem com o atributo *wordid*. Por fim, a tabela *imhistograms* contém o histograma de palavras visuais completo para cada imagem com o atributo *histogram*.

imlist
rowid
filename

imwords
imid
wordid
vocname

imhistograms
imid
histogram
vocname

Tabela 3.2: Esquema da base de dados SQLite

3.3 Matriz de distâncias entre imagens

Com o módulo descrito na secção anterior 3.2, a informação que descreve cada uma das imagens armazenadas localmente está alojada e é acessível através da base de dados local criada. Esta informação está no formato de um histograma que descreve a imagem através de um vocabulário visual. Este histograma é um vetor de tamanho único e igual para todas imagens, o que permite uma fácil comparação entre eles, o que é equivalente a dizer que estes permitem calcular a distância entre eles. Como cada histograma está associado a uma imagem, calcular a distância entre dois histogramas equivale a calcular a distância entre duas imagens.

O TweeProfiles utiliza o algoritmo DBSCAN apresentado no Capitulo 2 na secção 2.1.3 para realizar a tarefa de *clustering* ao conteúdo e às dimensões espacial e temporal. O *clustering* baseado em densidades, neste caso mais concreto, com o recurso ao algoritmo DBSCAN, pode utilizar uma matriz distância entre objetos para realizar a tarefa de *clustering*, sendo que retorna um número indefinido de *clusters* dependentes dos dados. A matriz distância trata-se apenas de uma matriz NxN em que N é o número total de objetos. No nosso caso, os objetos são as imagens e o valor de N é igual a 5958. A figura 3.3 apresenta a estrutura de uma matriz distância, em que 0 é a distância de um objeto a ele mesmo.

$$\begin{bmatrix} 0 & & & & \\ d(2,1) & 0 & & & \\ d(3,1) & d(3,2) & 0 & & \\ \vdots & \vdots & \vdots & & \\ d(n,1) & d(n,2) & \cdots & \cdots & 0 \end{bmatrix}$$

Figura 3.3: Estrutura de uma matriz de distâncias. *Retirada de* [7]

Tendo existido algumas limitações a nível de *hardware* para o calculo de uma matriz desta dimensões, optou-se por utilizar o mesmo procedimento utilizado no TweeProfiles [9] e dividir os dados em três partes distintas. Esta divisão foi efetuada tendo em conta uma divisão temporal, isto é, foram em primeiro lugar ordenados os tweets por ordem cronológica e apenas posteriormente foi divido em três partes iguais de 1986 tweets cada conjunto. Após este processo foi então calculadas as três matrizes.

Para calcular as matriz de distâncias utilizou-se o histograma descritor de cada imagem, calculando a distância entre histogramas através da função distância euclidiana apresentada no capítulo 2 na secção 2.1.5.

Com as matrizes distâncias calculadas, estavam as condições reunidas para a integração do modelo no TweeProfiles. O próximo capítulo irá introduzir a integração deste modelo com o TweeProfiles e o desenvolvimento da ferramenta Olhó-passarinho e os seus resultados ilustrativos.

Capítulo 4

Olhó-passarinho

Neste capitulo será abordado a ferramenta desenvolvida com a descrição da arquitetura sistema implementado, do processamento da informação espaço-temporal e da sua integração com a informação visual de modo a aplicar a tarefa de *clustering*. Por fim será apresentado a visualização dos resultados ilustrativos e consequentemente a sua discussão.

4.1 Arquitetura do sistema

A arquitetura do sistema desenvolvido é apresentado na figura 4.1. Este apresenta uma divisão entre os serviços externos e o modelo desenvolvido. Este modelo foi desenhado de modo a que existisse uma separação entre o tratamento de toda a parte de processamento dos dados e a visualização, existindo assim um *back-end* com todos os ficheiros e módulos desenvolvidos e um *front-end* que representa a aplicação web para visualização dos resultados. No *back-end* existe também uma divisão entre dois módulos fundamentais, o módulo de processamento da informação visual, responsável por tratar a informação das imagens como descrito no Capítulo 3, de modo a que essa informação possa ser utilizada pelo módulo responsável pelo processo de *Data Mining* já desenvolvido no TweeProfiles [9].

Os serviços externos correspondem à base de dados Mongodb para a recolha dos tweets e os serviços Twitter e Instagram para a recolha das imagens através do URL. No caso do modelo desenvolvido, a parte de *back-end* possuí os ficheiros JSON com os dados e as imagens necessárias, tanto para o módulo de processamento da informação visual como para a extração e processamento do dados espaço-temporais, explicados na próxima secção 4.2. Os dados processados no módulo da informação visual e os dados espaço-temporal extraídos dos tweets são assim utilizados no processo de *Data Mining*, onde é aplicada a tarefa de *clustering* como explicado na secção 4.3 apresentada mais adiante. De este processo resultam os *clusters* calculados através de vários parâmetros, sendo estas informações armazenadas em ficheiros. Por fim, foi utilizada a *microframework* Flask para desenvolvimento de aplicações web em Python, que permitiu o desenvolvimento da aplicação Olhó-passarinho para visualização dos resultados.

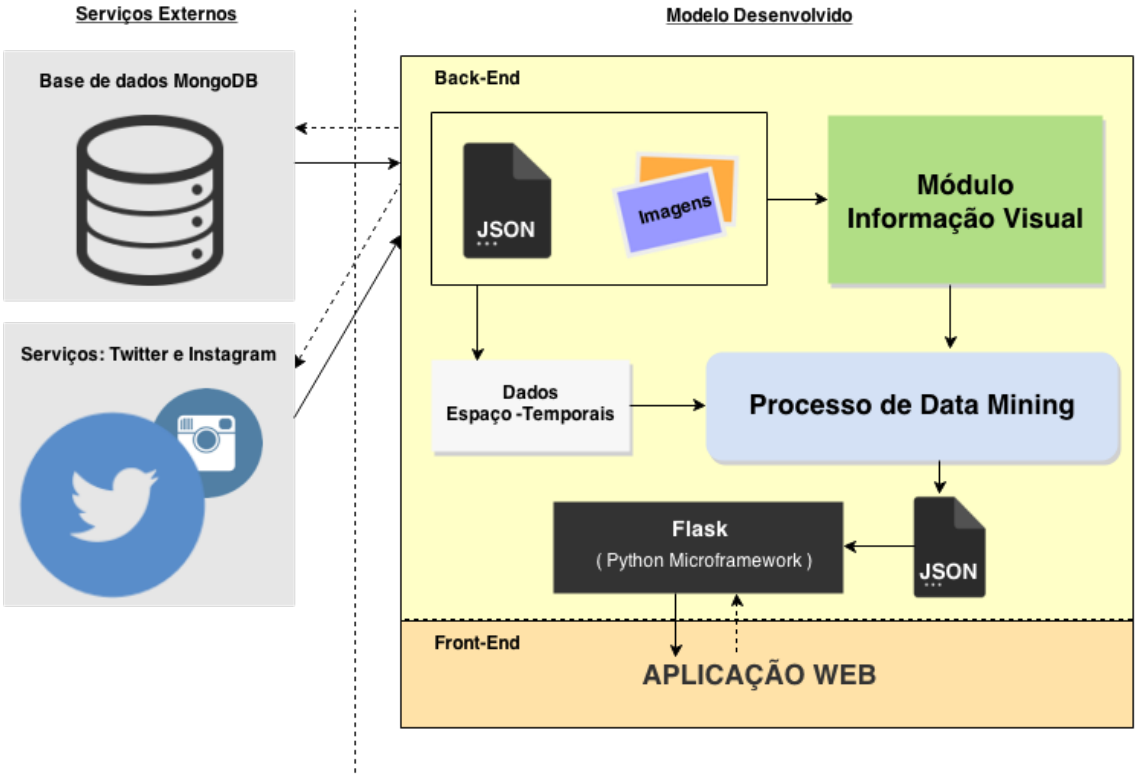


Figura 4.1: Arquitetura do sistema completo

4.2 Informação espaço-temporal

Uma das características principais tanto do TweeProfiles como do Olhó-paddarinho e é integração das dimensões espaço-temporais com o conteúdo. Tal como no foi efetuado no TweeProfiles, também aqui foi utilizado estas dimensões, tendo sido então recolhido a informação de tempo e espaço dos tweets para o cálculo das respetivas matrizes de distância entre tweets.

Em primeiro lugar foi recolhida a informação espacial. Neste caso os dados possuem a informação de latitude e longitude do ponto onde foi enviado o tweet. Para calcular a distância entre tweets utilizou-se a função distância Haversine abordada no capítulo 2 na subsecção 2.3.3.1. Neste caso foi calculada a distância em quilómetros, tendo sido considerado o valor do raio da Terra igual a 6371 Km.

Posteriormente foi então recolhida a informação temporal dos tweets. Esta informação apresenta-se no seguinte formato:

Tue Jun 18 17:02:09 +0000 2013

Para o cálculo da distância entre datas foi utilizada a função distância euclidiana, que se pode resumir ao módulo da diferença entre o tempo de dois tweets, como pode ser visto no capítulo 2 na subsecção 2.1.5.3. Utilizando as bibliotecas *datetime* e *dateutil* este cálculo é direto sem necessitar ser realizada uma conversão do formato da data.

Nota: Devo colocar mais informação estatística sobre os dados?

4.3 Clustering da informação visual, espacial e temporal

A tarefa de *clustering* é o passo final para a obtenção dos resultados finais. Este é o processo que engloba os dados resultantes de todo o processamento da informação tratada e discutida anteriormente.

Para a tarefa de *clustering* optou-se pela utilização do algoritmo implementado no TweeProfiles [9], o DBSCAN. Este apresenta algumas vantagens na utilização de dados recolhidos de redes sociais, pois este não necessita de uma predefinição do número de *clusters* que se pretende obter, sendo o número de *clusters* é definido através da distribuição em densidade dos objetos, como referido no capítulo 2 na subsecção 2.1.3.

Mas antes de fornecer os dados, neste caso as matrizes já calculadas com as distâncias entre tweets para as dimensões temporal, espacial e de conteúdo visual, foi necessário recorrer a sua normalização de modo a que fosse possível a combinação entre as várias matrizes. Foi então utilizada a normalização através da média e do desvio padrão.

Após a normalização, realizou-se a combinação entre as matrizes, com atribuição de vários pesos a cada matriz, de forma a que a soma dos diferentes pesos fosse igual a 1. Inicialmente, a distribuição dos pesos foi feita com passos de 0.25 pontos, isto é, os valores de cada matriz podiam assumir os pesos: 0, 0.25, 0.5, 0.75 e 1. Isto permitiu uma fácil divisão dos pesos, mas não permitia atribuir pesos iguais às três matrizes, pelo número de matrizes ser ímpar. Assim, optou-se por dividir os pesos em fator de 0.333, em que os valores de cada matriz podem assumir os pesos: 0, 0.333, 0.666, 1. Isto faz que a soma dos pesos posa não dar exatamente 1, mas sim 0.999. Por outro lado, isto possibilitou atribuir o mesmo peso às três matrizes diferentes de cada subconjunto.

A tabela 4.1 apresenta as diferentes combinações possíveis de pesos para as diferentes dimensões

Combinação	Visual	Espacial	Temporal
1	1.000	0.000	0.000
2	0.666	0.333	0.000
3	0.666	0.000	0.333
4	0.333	0.333	0.333
5	0.000	1.000	0.000
6	0.333	0.666	0.000
7	0.000	0.666	0.333
8	0.000	0.000	1.000
9	0.333	0.000	0.666
10	0.000	0.333	0.666

Tabela 4.1: Tabela com as possiveis distribuições de pesos entre as várias dimensões

4.4 Visualização

A aplicação web desenvolvida é a responsável pela visualização dos resultados obtidos pela tarefa de *clustering*. Esta apresenta três secções principais de visualização dos *clusters* pelas diferentes dimensões utilizadas.

O primeiro é um mapa, onde são apresentados a distribuição dos tweets, como pode ser visto na figura 4.2 e a respetiva distribuição dos *clusters* geograficamente. Este foi desenvolvido recorrendo à API Javascript do Google Maps v3 [33] disponibilizada pela própria Google, sendo toda ela controlada através da linguagem de programação Javascript. Esta permite utilizar um mapa e controlar de modo a adicionar componentes a esse mapa. Os tweets foram assim representados por pequenos círculos azuis, e os *clusters* como círculos vermelhos com transparência.

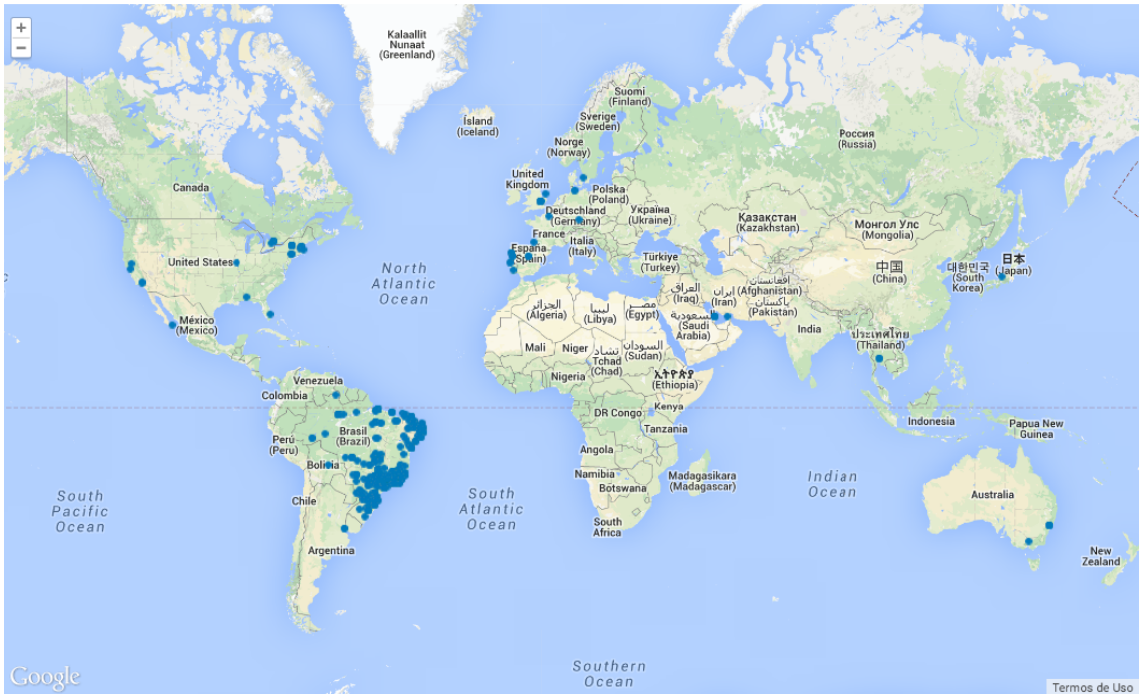


Figura 4.2: Distribuição de tweets na dimensão espacial

A segunda secção é responsável pela representação da distribuição dos *clusters* na dimensão temporal e foi utilizado a ferramenta Google Charts, mais especificamente a API Timeline [34]. Esta, tal como a API do Google Maps, também é desenvolvida em javascript, e permite reproduzir um gráfico com barras de duração temporal, como podemos ver na figura 4.3.

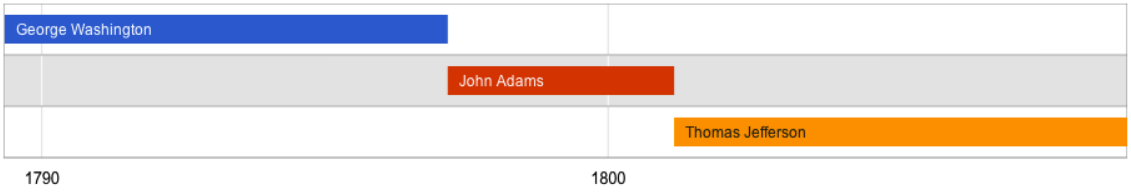


Figura 4.3: Exemplo ilustrativo da ferramenta Timeline. *Retirada de* [34]

Anexo A

Exemplo objeto JSON de um tweet


```
{
    "_id": {
        "$oid": "52c6d0f28ef20d397e42c54a"
    },
    "contributors": null,
    "coordinates": null,
    "created_at": "Tue Jun 18 17:08:15 +0000 2013",
    "entities": {
        "hashtags": [],
        "symbols": [],
        "urls": [
            {
                "display_url": "twitpic.com/cxvh38",
                "expanded_url": "http://twitpic.com/cxvh38",
                "indices": [
                    104,
                    126
                ],
                "url": "http://t.co/P7nwF6GOFc"
            }
        ],
        "user_mentions": [
            {
                "id": 1181030630,
                "id_str": "1181030630",
                "indices": [
                    3,
                    14
                ],
                "name": "Bruna",
                "screen_name": "holdmenian"
            }
        ]
    },
    "favorite_count": 0,
```

```
    "favorited": false,
    "filter_level": "medium",
    "geo": null,
    "id": 347038057221980162,
    "id_str": "347038057221980162",
    "in_reply_to_screen_name": null,
    "in_reply_to_status_id": null,
    "in_reply_to_status_id_str": null,
    "in_reply_to_user_id": null,
    "in_reply_to_user_id_str": null,
    "lang": "pt",
    "metadata": {
        "client": "192.168.102.195",
        "emoticons": [],
        "hashtags": [],
        "language": "pt",
        "mentions": [
            "@holdmenian"
        ],
        "tokenized": "RT @holdmenian : \" O comercial da fiat ' Vem pra rua '
    saiu do ar ap\u00f3s virar m\u00fasica tema dos protestos \" mas http://t.
    co/P7nwF6GOFc",
        "topic": "protestos_brasil",
        "urls": [
            "http://t.co/P7nwF6GOFc"
        ]
    },
    "place": null,
    "possibly_sensitive": false,
    "retweet_count": 0,
    "retweeted": false,
    "retweeted_status": {
        "contributors": null,
        "coordinates": null,
        "created_at": "Tue Jun 18 13:09:39 +0000 2013",
        "entities": {
            "hashtags": [],
            "symbols": [],
            "urls": [
                {
                    "display_url": "twitpic.com/cxvh38",
                    "expanded_url": "http://twitpic.com/cxvh38",
                    "indices": [
                        88,
                        110
                    ],
                    "url": "http://t.co/P7nwF6GOFc"
                }
            ],
```

```
            "user_mentions": []
        },
        "favorite_count": 7,
        "favorited": false,
        "geo": null,
        "id": 346978014636146688,
        "id_str": "346978014636146688",
        "in_reply_to_screen_name": null,
        "in_reply_to_status_id": null,
        "in_reply_to_status_id_str": null,
        "in_reply_to_user_id": null,
        "in_reply_to_user_id_str": null,
        "lang": "pt",
        "place": null,
        "possibly_sensitive": false,
        "retweet_count": 98,
        "retweeted": false,
        "source": "<a href=\"http://messaging.nokia.com/\" rel=\"nofollow\">
    Social by Nokia</a>",
        "text": "\"O comercial da fiat 'Vem pra rua' saiu do ar ap\u00f3s virar
     m\u00fasica tema dos protestos\" mas http://t.co/P7nwF6GOFc",
        "truncated": false,
        "user": {
            "contributors_enabled": false,
            "created_at": "Fri Feb 15 03:50:19 +0000 2013",
            "default_profile": false,
            "default_profile_image": false,
            "description": "make a wish \u221e",
            "favourites_count": 12,
            "follow_request_sent": null,
            "followers_count": 934,
            "following": null,
            "friends_count": 718,
            "geo_enabled": false,
            "id": 1181030630,
            "id_str": "1181030630",
            "is_translator": false,
            "lang": "en",
            "listed_count": 0,
            "location": "Charlie \u2665",
            "name": "Bruna",
            "notifications": null,
            "profile_background_color": "FFFFFF",
            "profile_background_image_url": "http://a0.twimg.com/
    profile_background_images/344918034409728850/6
    c945a8006333f3847476148aa68d7a0.png",
            "profile_background_image_url_https": "https://si0.twimg.com/
    profile_background_images/344918034409728850/6
    c945a8006333f3847476148aa68d7a0.png",
```

```
            "profile_background_tile": false,
            "profile_banner_url": "https://pbs.twimg.com/profile_banners
    /1181030630/1371263818",
            "profile_image_url": "http://a0.twimg.com/profile_images
    /344513261580011975/f105a548f1b2198d325864dc5313e06b_normal.png",
            "profile_image_url_https": "https://si0.twimg.com/profile_images
    /344513261580011975/f105a548f1b2198d325864dc5313e06b_normal.png",
            "profile_link_color": "B40B43",
            "profile_sidebar_border_color": "FFFFFF",
            "profile_sidebar_fill_color": "DDEEF6",
            "profile_text_color": "333333",
            "profile_use_background_image": true,
            "protected": false,
            "screen_name": "holdmenian",
            "statuses_count": 9291,
            "time_zone": "Mid-Atlantic",
            "url": null,
            "utc_offset": -7200,
            "verified": false
        }
    },
    "source": "web",
    "text": "RT @holdmenian: \"O comercial da fiat 'Vem pra rua' saiu do ar ap\
    u00f3s virar m\u00fasica tema dos protestos\" mas http://t.co/P7nwF6GOFc",
    "truncated": false,
    "user": {
        "contributors_enabled": false,
        "created_at": "Wed Mar 02 16:18:50 +0000 2011",
        "default_profile": false,
        "default_profile_image": false,
        "description": ".. o fim virou come\u00e7o. E eu me permiti come\
    u00e7ar.",
        "favourites_count": 6,
        "follow_request_sent": null,
        "followers_count": 405,
        "following": null,
        "friends_count": 360,
        "geo_enabled": true,
        "id": 259792830,
        "id_str": "259792830",
        "is_translator": false,
        "lang": "pt",
        "listed_count": 4,
        "location": "Pau dos ferros - RN",
        "name": "Fernando Cassio",
        "notifications": null,
        "profile_background_color": "759AAD",
        "profile_background_image_url": "http://a0.twimg.com/
    profile_background_images/396888075/bg-meio.jpg",
```

```
        "profile_background_image_url_https": "https://si0.twimg.com/
    profile_background_images/396888075/bg-meio.jpg",
        "profile_background_tile": true,
        "profile_banner_url": "https://pbs.twimg.com/profile_banners
    /259792830/1361109491",
        "profile_image_url": "http://a0.twimg.com/profile_images/3687937981/
    bd4525bc45aa159a1c8dabcb5eab3ef4_normal.jpeg",
        "profile_image_url_https": "https://si0.twimg.com/profile_images
    /3687937981/bd4525bc45aa159a1c8dabcb5eab3ef4_normal.jpeg",
        "profile_link_color": "888E94",
        "profile_sidebar_border_color": "D1D3DE",
        "profile_sidebar_fill_color": "E2EEF0",
        "profile_text_color": "131314",
        "profile_use_background_image": true,
        "protected": false,
        "screen_name": "fernandocassio_",
        "statuses_count": 5296,
        "time_zone": "Santiago",
        "url": "http://www.facebook.com/fernando.cassio.948",
        "utc_offset": -14400,
        "verified": false
    }
}
```

Referências

[1] Max Bramer. *Principles of Data Mining*. 2007.

[2] Peng Wu, YM Ro, CS Won, e Yanglim Choi. Texture descriptors in MPEG-7. *Comput. Anal. Images* . . . , páginas 21–28, 2001.

[3] M. Bober. MPEG-7 visual shape descriptors. *IEEE Trans. Circuits Syst. Video Technol.*, 11(6):716–719, Junho 2001. doi:10.1109/76.927426.

[4] David G. Lowe. Distinctive Image Features from Scale-Invariant Keypoints. *Int. J. Comput. Vis.*, 60(2):91–110, Novembro 2004. URL: http://link.springer.com/10.1023/B:VISI.0000029664.99615.94, doi:10.1023/B:VISI.0000029664.99615.94.

[5] Herbert Bay, Tinne Tuytelaars, e Luc Van Gool. Surf: Speeded up robust features. *Comput. Vision–ECCV 2006*, 2006. URL: http://link.springer.com/chapter/10.1007/11744023_32.

[6] LM Bueno. Análise de descritores locais de imagens no contexto de detecção de semi-réplicas. 2011. URL: http://www.dca.fee.unicamp.br/~dovalle/recod/works/lucasBueno2001mscDissertation.pdf.

[7] Jiawei Han, Micheline Kamber, e Jian Pei. *Data Mining, Second Edition: Concepts and Techniques*. 2006.

[8] Matthew A Russell. *Mining the Social Web*, volume 54. 2011. doi:10.1081/E-ELIS3-120043522.

[9] TDS Cunha. *TweeProfiles: detection of spatio-temporal patterns on Twitter*. Tese de doutoramento, Faculdade de Engeneharia da Universidade do Porto, 2013. URL: http://paginas.fe.up.pt/~ei08142/files/mieic_en.pdf.

[10] M Boanjak e Eduardo Oliveira. TwitterEcho: a distributed focused crawler to support open research with twitter data. *Proc. 21st* . . . , 2012.

[11] Dr. Matthew A North. *Data Mining for the Masses*. Global Text Project, 2012.

[12] Usama Fayyad, Gregory Piatetsky-shapiro, e Padhraic Smyth. From Data Mining to Knowledge Discovery in Databases. páginas 37–54, 1996.

[13] Bing Liu. *Web Data Mining: Exploring Hyperlinks, Contents, and Usage Data*. Springer, 2011.

[14] Wei Wang, Jiong Yang, e Richard R. Muntz. STING: A Statistical Information Grid Approach to Spatial Data Mining. páginas 186–195, Agosto 1997.

[15] Rakesh Agrawal, Johannes Gehrke, Dimitrios Gunopulos, e Prabhakar Raghavan. Automatic subspace clustering of high dimensional data for data mining applications. *ACM SIGMOD Rec.*, 27(2):94–105, Junho 1998. doi:10.1145/276305.276314.

[16] David A. Forsyth e Jean Ponce. *Computer Vision: A Modern Approach*. Pearson Education, Limited, 2011.

[17] Mark S. Nixon e Alberto S. Aguado. *Feature Extraction and Image Processing*. 2002.

[18] BS Manjunath e JR Ohm. Color and texture descriptors. *Circuits Syst. ...*, 11(6):703–715, 2001.

[19] Charilaos Christopoulos, Daniel Berg, e Athanassios Skodras. The colour in the upcoming MPEG-7 standard. *Invit. Pap. Eur. ...*, páginas 1–4, 2000.

[20] Leszek Cieplinski. MPEG-7 Color Descriptors and Their Applications. 7:11–20, 2001.

[21] Leszek Cieplinski (mitsubishi Electric Ite-vil. The MPEG-7 Color Descriptors Jens-Rainer Ohm (RWTH Aachen, Institute of Communications Engineering).

[22] Vinay Modi. Color descriptors from compressed images. 2008.

[23] H Shao, J Ji, Y Kang, e H Zhao. Application Research of Homogeneous Texture Descriptor in Content-Based Image Retrieval. *Inf. Eng. ...*, (2008515):2–5, 2009.

[24] Kristen Gauman e Bastian Leibe. *Visual Object Recognition*. Morgan & Claypool Publishers, 2010. URL: http://books.google.com/books?id=fYZgAQAAQBAJ&pgis=1.

[25] D.G. Lowe. Object recognition from local scale-invariant features. *Proc. Seventh IEEE Int. Conf. Comput. Vis.*, páginas 1150–1157 vol.2, 1999. URL: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=790410, doi:10.1109/ICCV.1999.790410.

[26] Josef Sivic e Andrew Zisserman. Video Google: A text retrieval approach to object matching in videos. *Comput. Vision, 2003. Proceedings. ...*, (Iccv):2–9, 2003.

[27] Josef Sivic e Andrew Zisserman. Video Google: Efficient visual search of videos. *Towar. Categ. Object Recognit.*, 4170:127–144, 2006. doi:10.1007/11957959_7.

[28] Luo Juan e O Gwun. A comparison of sift, pca-sift and surf. *Int. J. Image Process.*, (4):143–152, 2009. URL: http://www.cscjournals.org/csc/manuscript/Journals/IJIP/volume3/Issue4/IJIP-51.pdf.

[29] R Baeza-Yates e B Ribeiro-Neto. *Modern information retrieval*. 1999. URL: ftp://mail.im.tku.edu.tw/seke/slide/baeza-yates/chap10_user_interfaces_and_visualization-modern_ir.pdf.

[30] D. Nister e H. Stewenius. Scalable recognition with a vocabulary tree. *... Vis. Pattern Recognition, 2006 ...*, 2, 2006. URL: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1641018.

[31] Jan Erik Solem. Programming Computer Vision with Python. 2012.

[32] A. Vedaldi e B. Fulkerson. VLFeat: An open and portable library of computer vision algorithms. http://www.vlfeat.org/, 2008.

[33] Google. Api javascript do google maps v3. https://developers.google.com/maps/documentation/javascript/, 2013.

[34] Google. Google chart - timeline. https://developers.google.com/chart/interactive/docs/gallery/timeline, 2014.