Classificando imagens dos Simpsons

Leonardo Pontes Baiser[1], Marco Cezar Moreira de Mattos[1], Rômulo Manciola Meloca[1]

[1]DACOM – Universidade Tecnológica Federal do Paraná (UTFPR)
Caixa Postal 271 – 87301-899 – Campo Mourão – PR – Brazil

{lpbaiser, marco.cmm,rmeloca}@gmail.com

Abstract. *This report shows the process of developing a program whose goal is to apply on the basis of images the concepts of artificial intelligence seen in the classroom along with the concepts of literature, therefore discriminate five classes referring to characters in the TV series textit Simpsons, these classes extract features and starting these characteristics apply sorting algorithms.*

Resumo. *Este relatório mostra o processo de desenvolvimento de um programa cujo objetivo é aplicar sobre uma base de imagens os conceitos de inteligência artificial vistos em sala de aula juntamente com os conceitos da literatura, para tanto discriminamos cinco classes referentes aos personagens do seriado de TV Simpsons, nestas classes extraímos características e apartir destas características aplicamos algoritmos de classificação.*

1. Introdução

Muito se discute, sobre o tema Ïnteligência artificialë a utilização desta área para solucionar os inúmeros problemas existentes no mundo atual, embora a humanidade ainda engatinhe no desenvolvimento de soluções, alguns sistemas inteligentes já praticam o que chamamos de inteligência artificial, como é o casos de reconhecedores de padrões utilizando-se de imagens.

2. O Problema

Dado uma imagem de um banco de imagens dos Simpsons deseja-se saber qual é o personagem presente em cada imagem do conjunto de imagens, para tanto deve-se utilizar de extratores de características específicos para imagens, assim gerando conjuntos de características que a posteriori serão classificados utilizando classificadores do tipo K-NN (k-nearest neighbors), SVM (support vector machine) e DT (decision tree).

3. Organização da Solução

Para implementação da solução do problema a priori observou-se os conjuntos de treino e teste do problema proposto afim de encontrar características que maior definem e distinguem uma imagem de outra. Tomando-se um problema onde o conjunto de imagens expressam cores que melhor definem os personagens, nós optamos por extratores de características quem trabalham melhor com coloração de imagem,

então dada a escolha dos extratores, definimos o primeiro o extrator de cores predominantes, cada imagem é iterada sobre sua matriz de cores e dela quantificamos as cores que aparecem e são condizentes com as cores presentes na vestimenta de cada personagem, ao final da iteração temos um *hashmap* onde cada chave é a cor da roupa de um personagem e seu valor a quantidade de *pixels* presente na matriz, com este *hashmap* pagamos a cor predominante.
O segundo extrator de característica é um descritor de contorno *(shape descriptor)*, com este método extraímos o contorno de uma imagem, obtemos uma matriz binária onde 1 define o contorno, para podermos utilizar este resultado no classificador extraímos o perímetro do contorno.
O terceiro extrator de característica também é classificado com descritor de cor, com histogramas de imagens temos um vetor de variação de tons de cores, para utilização do histograma convertemos a imagem para tons de cinza, em alguns testes utilizamos o vetor normalizado.
No quarto extrator aplicou-se uma transforma de *fourier*, neste extrator o vetor de valores gerado no histograma é enviado para a classe FFT do java que transforma este valores em números complexos, utilizamos este conjunto de números complexos para ... Escrever para que utilizamos!!!

3.1. Diagramação

4. Implementação

Dado as bibliotecas já implementadas, como mencionado na seção anterior, desenvolveu-se a solução com a linguagem de programação C.

Além das abstrações já citadas, para facilitar o desenvolvimento e torná-lo mais natural, tornando-o mais próximo ao conceito de objetos, utilizou-se apelidos característicos de objetos para estruturas já nomeadas (como por exemplo *Connection* ao invés de *connection_t*) e definições de métodos redundantes (como por exemplo *sendPackage()* em contraposição à *CONN_send()*). Tais fatores, permitiram o encapsulamento de algumas funcionalidades e possibilidade de fácil e rápida manutenção/refatoração do código.

Tendo sido mapeado os *includes* no diagrama de classes, facilmente pode-se visualizar os módulos que os programas *server* e *client* deveriam incluir, mantendo a coerência com a distância que as partes (naturalmente) deveriam ter, com excessão da interface que os conecta, os pacotes definidos na biblioteca *package.h* (que já conta com a inclusão da biblioteca *connection.h*).

Colaborou-se o código com o auxílio do controle de versões git, onde o integrante Rômulo responsabilizou-se pela implementação do arquivo *server.c* e das duas *threads requestHandler.c* e *worker.c*. O núcleo da aplicação foi desenvolvido de maneira conjunta e interativa em relação as partes. O integrante Marco responsabilizou-se por todo o arquivo *client.c* além do *upload* do arquivo em *multi-parts*.

5. Considerações Finais

Considera-se, por fim, que o desenvolvimento de um projeto que conta com várias *threads*, o conceito de produtor-consumidor, um *buffer* compartilhado onde

apresentam-se condições de corrida bem específicas, a integração de vários módulos além da comunicação inter-processos via *socket*, permite no mínimo alargar os conhecimentos e fixar o aprendizado de todos esses conceitos vastamente utilizados nas mais diversas aplicações atuais. Nesse espectro, salienta-se a importância de tal desenvolvimento e, sobretudo, a fase de projeto, que tanto agrega para a visualização panorâmica deste ponto em específico da disciplina de Sistemas Operacionais.

Conclui-se que aplicações de determinada escala demandam a produção de vários artefatos, que somente são produzidos após o fiel debruçar-se nas ideias e adiantar-se a respeito de todos os problemas que são solucionados e gerados a partir delas. Toma-se como proveito o pensar em soluções que possam ser integradas a demais programas e o pensar em abstrações e interfaces que possam ser escaladas e reutilizadas.

Exemplo de referência CC01a [1].

Referências

[1] Chih-Chung Chang and Chih-Jen Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.