Intelligent tool for visual data analysis

Bachelor's Thesis

Jorge Oses Grijalba

Double Major in Mathematics and Computer Science
Computer Science Faculty
Complutense University of Madrid

March 2019

Documento maquetado con TEXIS v.1.0.

Este documento está preparado para ser imprimido a doble cara.

Intelligent tool for visual data analysis

Memoria que presenta para optar al título de Doctor en Informática

Jorge Oses Grijalba

Dirigida por el Doctor

Tutor no definido. Usa \tutorPortada

Double Major in Mathematics and Computer Science
Computer Science Faculty
Complutense University of Madrid

March 2019

Al duque de Béjar
y
a tí, lector carísimo

I can't go to a restaurant and
order food because I keep looking
at the fonts on the menu.
Donald Knuth

Agradecimientos

> *A todos los que la presente vieren y entendieren.*
>
> Inicio de las Leyes Orgánicas. Juan Carlos I

Groucho Marx decía que encontraba a la televisión muy educativa porque cada vez que alguien la encendía, él se iba a otra habitación a leer un libro. Utilizando un esquema similar, nosotros queremos agradecer al Word de Microsoft el habernos forzado a utilizar LaTeX. Cualquiera que haya intentado escribir un documento de más de 150 páginas con esta aplicación entenderá a qué nos referimos. Y lo decimos porque nuestra andadura con LaTeX comenzó, precisamente, después de escribir un documento de algo más de 200 páginas. Una vez terminado decidimos que nunca más pasaríamos por ahí. Y entonces caímos en LaTeX.

Es muy posible que hubíeramos llegado al mismo sitio de todas formas, ya que en el mundo académico a la hora de escribir artículos y contribuciones a congresos lo más extendido es LaTeX. Sin embargo, también es cierto que cuando intentas escribir un documento grande en LaTeX por tu cuenta y riesgo sin un enlace del tipo "*Author instructions*", se hace cuesta arriba, pues uno no sabe por donde empezar.

Y ahí es donde debemos agradecer tanto a Pablo Gervás como a Miguel Palomino su ayuda. El primero nos ofreció el código fuente de una programación docente que había hecho unos años atrás y que nos sirvió de inspiración (por ejemplo, el fichero `guionado.tex` de TeXIS tiene una estructura casi exacta a la suya e incluso puede que el nombre sea el mismo). El segundo nos dejó husmear en el código fuente de su propia tesis donde, además de otras cosas más interesantes pero menos curiosas, descubrimos que aún hay gente que escribe los acentos españoles con el `\'{\i}`.

No podemos tampoco olvidar a los numerosos autores de los libros y tutoriales de LaTeX que no sólo permiten descargar esos manuales sin coste adicional, sino que también dejan disponible el código fuente. Estamos pensando en Tobias Oetiker, Hubert Partl, Irene Hyna y Elisabeth Schlegl, autores del famoso "The Not So Short Introduction to LaTeX 2$_\varepsilon$" y en Tomás

Bautista, autor de la traducción al español. De ellos es, entre otras muchas cosas, el entorno `example` utilizado en algunos momentos en este manual.

También estamos en deuda con Joaquín Ataz López, autor del libro "Creación de ficheros LaTeX con GNU Emacs". Gracias a él dejamos de lado a WinEdt y a Kile, los editores que por entonces utilizábamos en entornos Windows y Linux respectivamente, y nos pasamos a emacs. El tiempo de escritura que nos ahorramos por no mover las manos del teclado para desplazar el cursor o por no tener que escribir `\emph` una y otra vez se lo debemos a él; nuestro ocio y vida social se lo agradecen.

Por último, gracias a toda esa gente creadora de manuales, tutoriales, documentación de paquetes o respuestas en foros que hemos utilizado y seguiremos utilizando en nuestro quehacer como usuarios de LaTeX. Sabéis un montón.

Y para terminar, a Donal Knuth, Leslie Lamport y todos los que hacen y han hecho posible que hoy puedas estar leyendo estas líneas.

Abstract

aaaaaaaaaaaa

aaaaaaa

This document reflects my Bachelor's Thesis corresponding to the Double Degree in Mathematics and Computer Science, developed within the area of intelligent data analytics and 'Case Based Reasoning'. During the progress of the project, the principles applicable in any environment of data processing and the science behind it are explained generally and aimed to be usable in any kind of context by any user provided the right format of data. Nowadays, highly heterogeneous data collection and processing methods are employed in all industries, however the techniques employed to get useful information out of the data usually have a generalistic aim, and the work relevant to the field itself is often done manually. In this work we aim to provide an automated way to analyze information while taking into account information and techniques relevant to the field of the analysis. The objective of this Degree's Final Project is the development of a prototype capable of carrying this analysis while being able to learn based on user input. As a Proof of Concept, we have included several medical domains with each one having developed specific methods and techniques for them. To serve as a base for this analysis, we have also developed a system for storing, loading and analyzing the information of the domain and the information provided by the user. This system will be the backbone of our architecture and enable the Case Based Reasoning analysis to function correctly in very different situations, providing the metrics and functions needed for every case.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Abstract: In this chapter we outline the thesis, state what our objective is and what we hope to achieve and list the programming libraries, techniques and methods used to achieve our goal.

1.1 Overview

The logical structure chosen for the program reflects the need for our tool to be a fully functional agent in and out of itself. We have designed it with a clear divider between a backend capable of storing the information and handling at the lowest possible level, which provides the frontend side with easy methods to get the information it needs, which is then processed taking who is going to look at it into account and then adequately presented to the user. A cornerstone of the program's functionality is to be able to remember decisions taken by a certain user and to be able to compare new data to old data of its kind. From these two necessities it is natural to consider some kind of identification system for our datasets, as automated as possible so it needs minimum user input and remains independent of the use case. For our program, if two datasets contain the exact same set of column names then they are considered to be comparable to each other, and every information stored about this kind of datasets will carry an identifier with the column names. From here onwards, the term 'domain' shall refer to information coming from the same kind of dataset.

1.2 Workflow

When a new dataset doesn't match any previous knowledge, our program automatically creates a new representation for these datasets which is stored along the others. If it detects a matching JSON with knowledge o its domain it loads that instead. Each representation of a domain stores data such as how many datasets have been loaded and a number of stats for each dataset and its columns depending on its types which will be specified later. Also, each domain has a number of 'profiles' associated which correspond to *who* this data is associated with. These profiles contain both historical data of the specific owner of the data (in our practical example, the patient data), and who will watch the report generated by this program, that is, the user of the program. The information that we're using will be stored in a specific JSON format for each kind that will be specified in chapter 2. When a dataset is introduced, the program loads the previous information, analyzes it, compares it and generates relevant information to the user. Then it updates the information with both the results of the analysis and user provided information. This workflow will be the basic use case of the program for every kind of data.

1.3 Structure

A clear module structure is provided so each module does a task in the workflow. The main modules on the backend side of our application are the Storage module, the CBRStorage module and the Analysis module. For the frontend, the logic structure will be split into the Reporter module and the Presentation module.

1.4 Tools

Our programming language of choice has been Python, particulary making use of its class to dictionary representation methods which make the work of manipulating the JSON structures much easier than using more rigid languages. A public repository has been created at (TODO:link here), and we have used Jupyter Notebooks for the testing and formation o a prototype which has been then moved into standard Python packages.

Notas bibliográficas

These are the bibliographical notes (**?**)

En el próximo capítulo...

This is the next chapter section

Chapter 2

The program structure : Backend

Abstract: In this chapter we provide an analysis of the backend side of our tool, examining how it works and what was and was not designed for, and its interaction with the frontend side which will be explained later. We provide both an abstract analysis and a more low level code explanation.

2.1 The Structure of the Backend

As we have stated before, the backend side of our program consists of a series of modules designed to interact with each other and provide the necessary tools for our program to work smoothly regardless of the data being used. Each module has a main class which provides almost all of its functionality, along with some tests for us to make sure that it's working properly. Here we take a look at the main classes of each module and main JSON structures we use to store the information.

2.2 The JSON Storage Object

This object is used to represent the raw information stored.

It contains statistics and properties from both columns and datasets, different for different types, as well as how they were measured (by saving the function to be called on a pandas dataset in the case of a dataset stat, or on the values themselves in the case of a function stat).

Saving how they're measured is important to measure new datasets and to be able to compare metrics effectively.

Both the column specific stats and the dataset stats are subjective to change, and the knowledgedomain object can be modified and adapted to fit new parameters easily.

This structure is generated for a single dataset and then combined with the domain one to take account of the new one.

2.3 The Storage Module

This module interacts with the JSON storages, its main functions are to retrieve the appropiate domain, create it if it doesn't exist, update it when it's needed and pass it along to the analyzer module.

It has a *load* method, which takes a list of column names (or "attributes"), a path and an optional name for the domain in case it's new.

It searches the given directory for .json extensions and then opens each one, comparing its "attributelist" key with the one provided to the class itself. If they match, it loads into itself and then returns the contents of the JSON.

If there is no match, creates an empty object with the attribute list and no further stats, and returns that instead.

It also has a *modify* method, which takes a dictionary and overwrites the information stored in the class with the new information from that dictionary, and a *save* method, which simply saves the class representation to a JSON file, intended to be used at the end of the program cycle.

A function to print information has also been added as an utility method to this class.

Also it's worth mentioning we have the ability to extend this class so some of its methods have been designed with the intention of being reused in the Profile Storage module

2.4 The JSON Profile Storage Structure

Another kind of information is stored about the domains. This is the information concerning the **human** side of things, that is, **how** to interpret these stats and turn them into something that humans with different levels of familiarity can understand.

To do this, we provide use another storage class that will contain human-relevant data that will modify the objective comparison delivered by the analysis module.

A system of profiles is added to the object itself, inside a "profiles" key. The domain associated is clear as they share the same "attributelist" identification system.

The information contained in each one of these "profiles" serves two different purposes. It provides customizable elements of *how* the data will be presented to the user, and it keeps an historical record of this user's dataset results (similar to the one in the domain storage) making a historical following of a profile possible.

For each domain, there's a *default* profile. This provides a way to present the data when no previous knowledge of the profile is available. The automated processes of obtaining this profile and tuning the existing profiles from user feedback will be explained in further chapters.

2.5 The Profile Storage Module

This module makes use of the tools provided by our Storage Module, the class itself extends the Storage class providing the extra methods needed to use the "profile" system. It has the same methods as that class but also provides a way to change or load a single profile, its functioning mirrors the Storage

2.6 The Comparison Metrics

The purpose of these functions is to provide a way to measure the properties of a given dataset or knowledge domain. We can categorize them as follows: First the "measurement" metrics, used to get the information of a single dataset or domain.

- Dataset Metrics : they concern the dataset as a whole, like number of rows with missing values. dm :: (ds) –> num

- Single Column Metrics : they concern a certain column, and are based on the type of the column. For numerical columns we will have things like median, averages, deviations, distributions... For categorical columns we'll work with frequencies and things of the sort. scm :: (col) –> num

- Multiple Column Metrics : we will be looking for correlations and things of that sort. scm :: (col,col) –> num Time based metrics will be defined from this construct.

We will also have "comparison" metrics, used to compare datasets against their domains. These metrics will compare the output of two measurement metrics, both will have to spawn from the same function. compm :: (metric) –> num

Note that these metrics are not to provide "meaning" or any human-readable input, nor to be inherently comparable between each other outside of a framework of understanding of the domain (metric importance).

A mean to convert these machine cold metrics into human understanding will be provided in further modules. For now, we're not taking humans into account.

2.7 The Analysis Module

The analysis module will receive a dataset, use the Storage module to load its information, then analyze the dataset, which generates a similar object to the domain json, then producing a comparison of both.

The main methods for this module are *getstats*, *getcolumnstats* and *getdatasetstats*.

The *getstats* method is just a wrapper for the other two, calls them both and stores its results inside the Analyzer class.

Both *getcolumnstats* and *getdatasetstats* compute the statistics for the given dataset. If there is previous knowledge of the domain, the stats that appear there are computed for the domain. If not, a standard set of frequencies for categorical values and medians and distributions for numerical values are calculated and used to populate the stats object.

The most important method is *getanalysis*. Once the stats for the datasets have been generated, if there's previous knowledge available the class runs an analysis comparing the metrics of the two, and generating an object with the result. For this to happen, each metric defined for the dataset must have an associated *comparison metric*.

If there is no previous knowledge then the dataset stats are passed along to the reporter with a field indicating that there was no previous knowledge.

In any case, at this level we've already filtered what is relevant and what is not from the comparison.

2.8 The Report Generation Module

This module stands at the edge between the backend and the frontend. It receives the information from the comparison between the dataset and the domain knowledge and extracts the relevant profile information.

Once this is done, it uses both to generate a report with all the information from both sides. The user-relevant info will modify what is shown and *how* that is shown, changing the graphical elements according to the user so the frontend modules are able to be logic-free.

It is able to directly modify the profile information by the proxy methods *modify* and *savehumaninfo*. Its main method, *generate*, will create and populate an attribute within itself called report.

This method is called when the class itself is generated but the report can be modified as any Python attribute if needed.

At this point, we have an object that represents the dataset compared to the historical data and data about the profile associated with the user. This information, however, is in the form of a JSON object and is not really human-readable. The job of the frontend modules is to take this information and turn it into something easy to understand.

Notas bibliográficas

These are the bibliographical notes (?)

En el próximo capítulo. . .

This is the next chapter section

Chapter 3

The Program Structure : Frontend

Abstract: In this chapter we provide an analysis of the frontend side of our tool, examining how it works and what was designed for, aswell as its interaction with the backend side. We provide both an abstract analysis and a more low level code explanation.

3.1 The Structure of the Backend

In our design process we have tried to keep the frontend side logic-free, its only functions being presenting the information to the user and providing the functionality of updating the relevant information with input from the user side. The two most important modules are the frontend module, which takes a *report* object from the *reporter* module and turns it into text and relevant graphs, directly displayed to the user by the interaction module.

3.2 The Frontend Module

As we have explained in the previous chapter, this module will receive an abstract representation of a Report as a JSON object. Its main function, *show*, will take a report object from the *reporter* class and turn it into a visual report, which contains both descriptive text and graphical representation of the information which has been labelled as relevant for the user. Every decision as to how to represent the information has already been taken before, so this module can be taken as a sort of machine to human translator.

3.3 The Interaction Module

This module's main function will be to take user feedback as to what is not relevant or what they would like to see, and then automatically update the information in the profile storage module so it can be used in the future by the same user.

Notas bibliográficas

These are the bibliographical notes (?)

En el próximo capítulo...

This is the next chapter section

Chapter 4

CBR Application

Abstract: In this chapter we outline and explain the Case Based Reasoning methodology used to make our tool more relatable, and delve deeper into the human side of our tool, and how it interacts with the other parts. We provide both an abstract approach and the decisions taken to make the implementation.

4.1 Frontend

This is the introductory texttt

Notas bibliográficas

These are the bibliographical notes (?)

En el próximo capítulo...

This is the next chapter section

Chapter 5

CBR Application : User Input

Abstract: In this chapter we outline and explain the user side of the Case Based Reasoning methodology used to test our tool with humans and to better define its ideal workflow. We provide both an abstract approach and the decisions taken to make the implementation.

5.1 CBR Applications

This is the introductory texttt

Notas bibliográficas

These are the bibliographical notes (**?**)

En el próximo capítulo...

This is the next chapter section

Chapter 6

Proof Of Concept
Medical Data

ABSTRACT: In this chapter we see all of the pieces of our program come together and work in the analysis of a set of medical data. We will follow a step by step execution from the different user viewpoints and the program's viewpoint.

6.1 introduction

This is the introductory texttt

Notas bibliográficas

These are the bibliographical notes (**?**)

En el próximo capítulo...

This is the next chapter section

Appendix A

Así se hizo...

...

...

Abstract: ...

A.1 Introducción

...

–¿Qué te parece desto, Sancho? – Dijo Don Quijote –
Bien podrán los encantadores quitarme la ventura,
pero el esfuerzo y el ánimo, será imposible.

Segunda parte del Ingenioso Caballero
Don Quijote de la Mancha
Miguel de Cervantes

–Buena está – dijo Sancho –; fírmela vuestra merced.
–No es menester firmarla – dijo Don Quijote–,
sino solamente poner mi rúbrica.

Primera parte del Ingenioso Caballero
Don Quijote de la Mancha
Miguel de Cervantes