FRI Data Science Project Competition 2023
Final report

How to automatically compare two Power BI reports?

Ilija Tavchioski

Abstract

Power BI is one of the most popular software for creating and manipulating data reports. A common practical scenario is that the data underlying the report are migrated, while the structure of the reports remains the same. In such cases we want to verify that the migration was performed correctly. One approach that avoids checking all the data is to verify that the reports on migrated data are the same as those using the data before the migration. We implemented a solution that takes two Power BI reports as input and outputs an *Excel* report that shows where the underlying data are the same and where they are not.

Keywords

Power BI, Business Analytics, Software development, Data comparison

Advisor: prof. dr. Erik Štrumbelj

Introduction

Data migration is a common process for companies that use large amounts of data. During migration, the location, computing environment, or even the form the data are stored in may change and errors may occur. There are several methods to check whether the migration is done correctly, such as **Completeness check** where we iterate over each data instance and compare them, but this method is expensive with respect to time, **Data sampling** is when we check only a sample of the data but, since we do not have all the data checked, this could lead to some issues with the functions later on. Motivated by *In516ht*, we focus on the approach of checking the correctness of the migration by comparing two *Power BI* reports that are identical in structure, but one queries the migrated data and the other the data before the migration.

Our approach is based on automatically segmenting the reports into individual visualizations and creating an *Excel* report that shows the differences between each pair of corresponding visualizations. And based on the aforementioned approach we successfully developed a software that will meet the main objectives for the most used and common structures of *Power BI* reports.

Problem description

Given two *Power BI* reports that are structured identically, the goal is to automatically detect the visualization from every page of the reports, extract the data used for the visualization, compare the two data sets, and output an *Excel* report that shows the differences.

What is a Power BI report?

Power BI is software developed by *Microsoft*, which provides an interactive and dynamic visual representation of the data. It allows users to connect to their database, transform and model the queried data, and create interactive reports with it. These reports have many additional features, such as filtering, calculating measures, and slicing the data. In addition, these reports, once created locally can also be uploaded to the *Azure server* where they can be accessed by having the corresponding credentials along with the *workspace-id* and *report-id* of the report.

The main structure of the report, which is unalterable, is defined as follows: A navigational bar on the left side, a settings bar at the upper part, along with options for filtering on the right side. These parts are fixed and are not important for our case. The most important part is the body of the report where the visualizations are presented, and this part is variable and can be in various colours, forms, and structures.

Data

In516ht provided us with four pairs of reports generated using their data, where each pair has identical structure but potentially different underlying data. Visually, the background of each visualization was white and the background of the *Power BI* report was light gray colour, which improves readability.

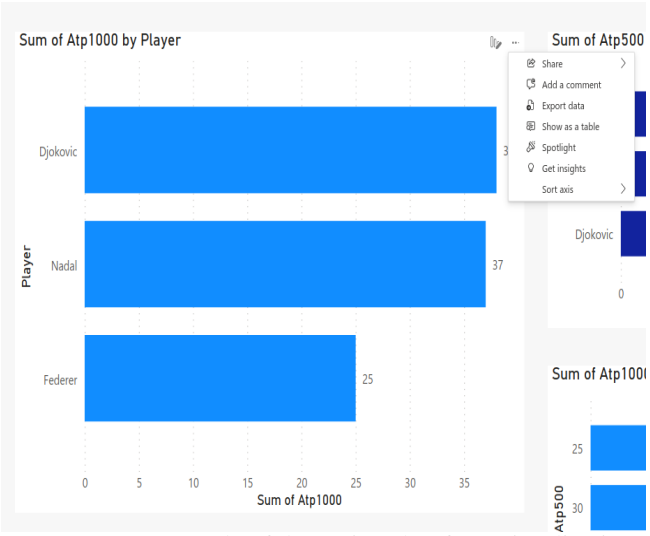


Figure 1. An example of the options bar for a visualization.

Data visualization

The underlying data can be visualized in different ways: text, statistical plot, matrix, table, hierarchy, etc. Every visualization is created from the calculations performed on a dataset that is queried from the database. The challenge here is that the querying of the dataset is not performed by *Power BI* and a comparison between the datasets would be expensive with respect to time consumption, which left us with the last option, to compare only that data that the visualizations were created from. Each visualization has a *More options* button, which can be shown while interacting with the corresponding visualization (typically in the right upper corner but sometimes in the right bottom corner). And, as can be seen in Figure 1, clicking the *More options* button shows a list of options. The *Export data* option, which when clicked, downloads the data used to create that visualization in CSV format. These data are the target of our comparison.

What counts as different?

If all rows are identical then the data are considered *the same*. Additionally, being the same is invariant to permutations of columns and/or rows. However, the number of rows (columns) must match or the data are considered *different*.

Once the data are matched by reordering the rows and/or columns, every difference counts as being different and should be reported, regardless of the language, currency, metric, etc. In addition, if the values at both tables are numerical, their difference in value should be reported.

Software and algorithms

We developed our solution in *Python* and it is composed of two main parts data extraction and data comparison. **Data extraction** inputs two *Power BI* reports uploaded to the *Azure server*, along with the required credentials, and outputs a list of tables representing the data used for the visualizations.

Data comparison compares the data and generates a report in *Excel*. The software has a command line interface with the following arguments: **username** – the username of the user's account on *Azure server*, **password** – the user's password, **gropuid1** – the *id* of the workspace where the first report is uploaded, *reportid1* – the *id* of the first report, **gropuid2** – the *id* of the workspace where the second report is uploaded, *reportid2* – the *id* of the second report, **data-folder** – the path to the folder where the data will be stored.

Data extraction

For the first part, we tried several approaches to extract the data. One of them was by using the *Power BI API* which was developed by *Microsoft* for interacting with the server where the *Power BI* reports are stored. Unfortunately, the API's functionalities are not sufficient, since it does not provide information about each visualization separately, but only with the dataset used for the whole report. The data for each visualization is calculated from that dataset, but the API does not provide the calculation formula. Another approach would be by extracting directly from the *Power BI* file (PBIX). But, this approach is more complicated since Power BI queries the data every time it starts, so the data are stored in memory and removed when we close the report, which would also require manual interaction. To extract the data we implemented a crawler using *Selenium*, where we detect the position of each visualization at each page of the report and click on the *More options* button in order to finally export the needed data for comparison. In addition, since the button is not always at the upper right corner, we also have the functionality of the crawler to try to find the button at the bottom right corner, if the button is not found at these two positions, it is considered that this is not a visualization that is created from data and thus not our objective for comparison.

The main problem with this part of the software was how to automatically locate each visualization of the report. In order to solve the aforementioned challenge we developed an algorithm that is using image processing methods on a screenshot of the *Power BI* report.

An important assumption is that the background of each visualization will always be white, and the background of the *Power BI* report will always be of some colour that has high brightness. The main reason that we opted for this approach is the fact that most useful reports give more importance to the charts, pies, plots, and tables and they are usually with darker colours and in contrast to the background of the visualization, and often the colour considered to be as a background is white. Regarding the background of the report, usually, people who work with data analyses do not give too much importance to the background color and they usually left to the default light gray colour, since using much darker colours would give unnecessary high contrast.

Data comparison

Once the data are downloaded, we must check if a reordering exists so that the data match. We developed the following

algorithm. With the described greedy approach, we sorted the instances based on how similar they are with respect to how many values are matched, and the result is the best possible matching between the tables without knowing their key values. After the rearrangement of the data instances, we simply compare each instance of the data between each visual where we defined several possible outcomes that we interpret as differences.

- Only one value is a string.
- Both values are strings, but they are different.
- One of the values is undefined.
- Different values and/or currency.

Everything else is considered to as an identical value. Another aspect here is the definition of a currency, here we defined the value to be a currency where the currency sign is either immediately before the numerical value or to be concatenated after the numerical number with one space (' '), everything else is interpreted as a string value or a mistake in the defining the value with currency.

Algorithm 1: Localization of the visualizations.

1. Get a screenshot of the report using the *Selenium* library.
2. Convert the image to a grayscale image.
3. Change each pixel that has a value lower than 240 to value 255 (white colour).
4. Change the rest values that are not 255 to black. At this point we have an image of only black and white pixels.
5. Find the contours in the image by using an algorithm proposed by Suzuki S. [1] which was implemented at the *OpenCV* [2] library.
6. By using the Douglas–Peucker algorithm that will connect the segments if they formed a line, we extract all possible rectangles from the image.
7. All rectangles that are close to the edges of the image are removed, along with rectangles that are too small.
8. Finally, we return for each rectangle its (x, y) position along with its width and height.

Results output

The result that we focused on was an *Excel* report Fig. 3 that will have the same number of pages as the *Power BI* report with an additional page which will represent the *Overview* Fig. 2 comparison between each pair of visualizations from the reports. In addition, it is implemented a link to the comparison in the report for each pair of visualizations, which links to the page with the detailed comparison. Then, on each page for every pair of visualizations, the data used for creation is presented, along with the corresponding number of rows and columns and their differences. In addition, the cells showing the differences are coloured red (different) or green (the same). The code and instructions on how to use it can be found here: https://github.com/theteleton/project.

Algorithm 2: Rearrangement of the data instances

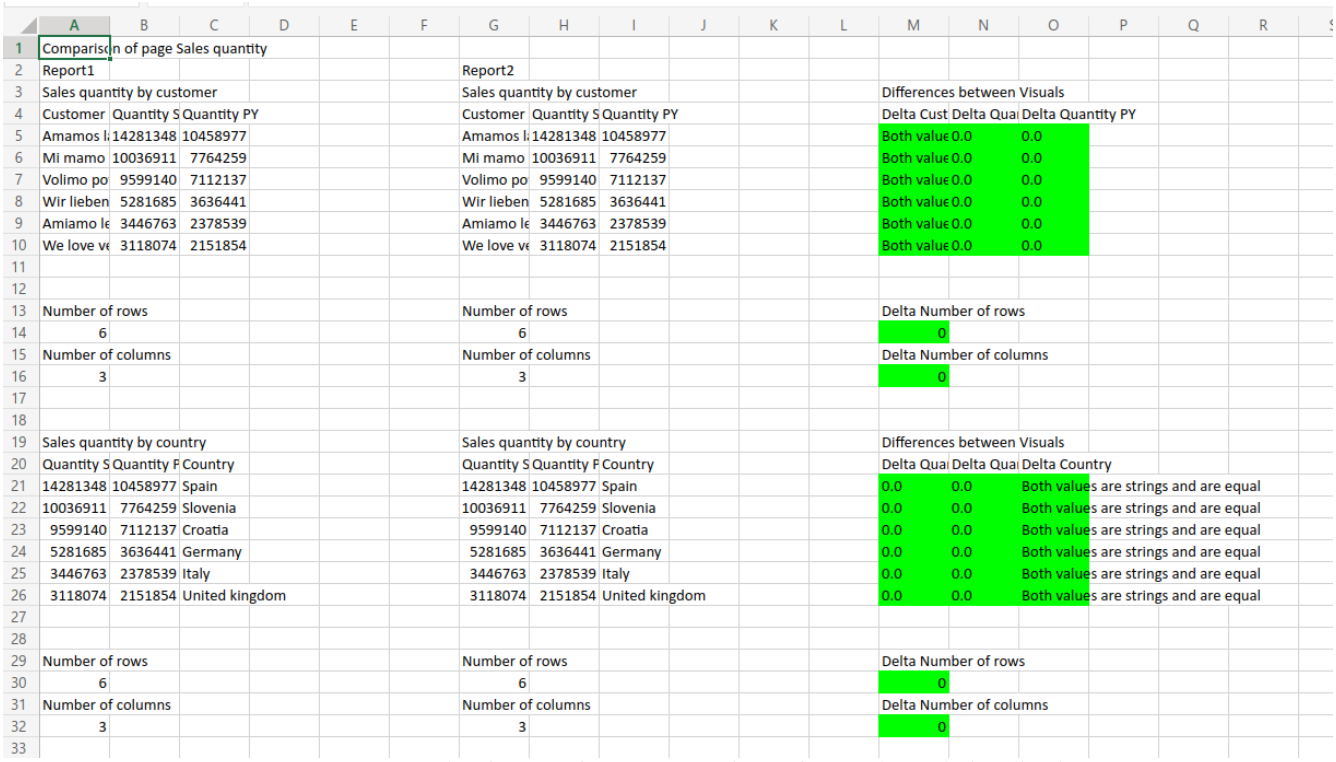
1. Initialize two new tables.
2. Rearrange the columns in both tables in such way that, first are the columns that are mutual and then the rest.
3. Find the pair of instances (one from the first and another from the second table), that have the most values matched between them.
4. Add both instances to the new tables.
5. Repeat the process from 2 to 4 until one of the tables is empty.
6. Move the left data instances to the bottom of the corresponding new table.

	A	B	C
1	Overview		
2	Page	Visual	Status
3	Sales quantity	Sales quantity by customer	Match
4	Sales quantity	Sales quantity by country	Match
5	Sales quantity	Sales quantity by veggie	Match
6	Sales quantity	Quantity SY by Veggie	Match
7	Sales quantity	Quantity SY by Country	Match
8	Sales value	Sales value by customer	Match
9	Sales value	Sales value by country	Match
10	Sales value	Sales value	Match
11	Sales value	Details	Match
12	Detailed view	Details value	Match
13	Detailed view	Details quantity	Not Match
14			

Figure 2. The *Overview* part of the comparison report.

Discussion

We achieved the main objective of our work. We implemented software that **automatically** locates each visualization on every page from the report and is generating a comparison report showing the differences between each pair of visualizations. As far as we are aware, this is the first software that does this.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
1	Comparison of page Sales quantity																	
2	Report1						Report2											
3	Sales quantity by customer						Sales quantity by customer						Differences between Visuals					
4	Customer	Quantity S	Quantity PY				Customer	Quantity S	Quantity PY				Delta Cust	Delta Qua	Delta Quantity PY			
5	Amamos l	14281348	10458977				Amamos l	14281348	10458977				Both value	0.0	0.0			
6	Mi mamo	10036911	7764259				Mi mamo	10036911	7764259				Both value	0.0	0.0			
7	Volimo po	9599140	7112137				Volimo po	9599140	7112137				Both value	0.0	0.0			
8	Wir lieben	5281685	3636441				Wir lieben	5281685	3636441				Both value	0.0	0.0			
9	Amiamo le	3446763	2378539				Amiamo le	3446763	2378539				Both value	0.0	0.0			
10	We love ve	3118074	2151854				We love ve	3118074	2151854				Both value	0.0	0.0			
11																		
12																		
13	Number of rows						Number of rows						Delta Number of rows					
14	6						6						0					
15	Number of columns						Number of columns						Delta Number of columns					
16	3						3						0					
17																		
18																		
19	Sales quantity by country						Sales quantity by country						Differences between Visuals					
20	Quantity S	Quantity F	Country				Quantity S	Quantity F	Country				Delta Qua	Delta Qua	Delta Country			
21	14281348	10458977	Spain				14281348	10458977	Spain				0.0	0.0	Both values are strings and are equal			
22	10036911	7764259	Slovenia				10036911	7764259	Slovenia				0.0	0.0	Both values are strings and are equal			
23	9599140	7112137	Croatia				9599140	7112137	Croatia				0.0	0.0	Both values are strings and are equal			
24	5281685	3636441	Germany				5281685	3636441	Germany				0.0	0.0	Both values are strings and are equal			
25	3446763	2378539	Italy				3446763	2378539	Italy				0.0	0.0	Both values are strings and are equal			
26	3118074	2151854	United kingdom				3118074	2151854	United kingdom				0.0	0.0	Both values are strings and are equal			
27																		
28																		
29	Number of rows						Number of rows						Delta Number of rows					
30	6						6						0					
31	Number of columns						Number of columns						Delta Number of columns					
32	3						3						0					
33																		

Figure 3. An example of a page from a comparison of several equal visualizations.

Since it is fully automated the software without any human assistance can process as much pair of reports as possible in a respected time span and generate for each pair an comparison report. This functionality is also parallelizable, which can be further useful it can process pairs much faster than a human expert could. Processing a pair is also trivially parallelizable.

Our solution has limitations. **That it increases linearly with number of visualizations is not a limitation, it has to increase at least linearly, because at a minumum, we have to go through each visualization at least once.** The assumptions about the visualization and background colours could be lifted by developing a more robust segmentation algorithm, most likely based on deep neural networks. Rearrangement of the data has time complexity of $O(n^3)$, where n is the number of instances, which would be a problem for large n. If we calculate beforehand, the maximum number of matching between the instances, we can stop the search when we find the best match. Assuming that most instances will be matched as best as possible most of the time, we can improve the complexity to quadratic.

addition, there might be also a way to bypass the need to click on each *Export data* button to get the data by getting the data at the moment when they are loaded at memory (when using the Power BI desktop software). But this method would require more research and knowledge of the architecture of the *Power BI* software.

Acknowledgments

I would like to thank the **DataScience@UL-FRI** initiative for providing the Project course as part of the Data Science Master's program. I would also like to thank **In516ht** and especially Vid Smrke for taking part in the *Project Competition* and providing data and advice. And finally, I want to thank prof. dr. Erik Štrumbelj for the guidance and responsiveness during the competition.

References

[1] Satoshi Suzuki and Keiichi Abe. Topological structural analysis of digitized binary images by border following. *Computer Vision, Graphics, and Image Processing*, 30(1):32–46, 1985.

[2] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.