

D3.7 Technical Brief on **AGGREGATOR**

WP3 Actorness and influence of the EU
in home and global governance

Grant Agreement n° 822735, Research and Innovation Action



This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement n° 822735. This document reflects only the author's view and the Commission is not responsible for any use that may be made of the information it contains.



TRIGGER

TRends in Global Governance and Europe's Role

Deliverable number:	
Deliverable name:	D3.7 Technical Brief on AGGREGATOR
WP / WP number:	WP3 Actorness and influence of the EU in home and global governance
Delivery due date:	M20 (31.07.2020)
Actual date of submission:	31.07.2020
Dissemination level:	Public
Lead beneficiary:	CEPS
Contributor(s):	Moritz Laurer (CEPS) Gaby Umbach (EUI)

Changes with respect to the DoA

-

Dissemination and uptake

Public

Evidence of accomplishment

Report

Content

1. Introduction	1
2. AGGREGATOR - Overview	1
3. Technical Development of AGGREGATOR	2
4. Technical details on embedded data visualisations.....	3
4.1. Back end – Hosting via Amazon Web Services (AWS)	3
4.2. Front end – Visualisation with Dash.....	4
5. Content of AGGREGATOR	5
6. Conclusion	6
Annex 1. Eurlex app code example	7

Index of Tables

Table 1 - AGGREGATOR data repository	5
--	---

1. Introduction

This deliverable provides an overview of the technical development of AGGREGATOR. Chapter 2 provides a general overview of AGGREGATOR and its functionalities. The following chapters describe the technical development of AGGREGATOR (chapter 3) as well as technical details on the front end and back end of the embedded data visualisations (chapter 4). Chapter 5 provides an overview of the different resources that will be available on AGGREGATOR. Please note that this deliverable is a technical brief on the software and technology used for AGGREGATOR whose full disclosure in some parts (i.e. detailed developer code) is subject to copyright restrictions. The present deliverable D3.7 therefore provides an overview on key details of this technical development. Conceptual details from a social science perspective are part of other deliverables.

2. AGGREGATOR - Overview

AGGREGATOR stands for '*Atlas of Global Governance REGulation and Europe's AcTORness*'. The TRIGGER project develops this Atlas in several steps. It is an online project database, that incorporates the datasets on global and EU governance developed in TRIGGER. At a later stage of the TRIGGER project, it will be embedded in another project development, the PERSEUS online tool (see Point 5).

The technical development of AGGREGATOR is the implementation of the following TRIGGER deliverables:

- Database on existing global governance regimes, tools and approaches, i.e. software development for open data access as input of AGGREGATOR (D3.3)
- Database on measuring global governance instruments, i.e. software development for open data access (D3.4)
- Online atlas of the EU's involvement in global governance, i.e. software development for open data access (D3.5)
- Technical Brief on AGGREGATOR (D3.7).

TRIGGER AGGREGATOR has been set up as an independent software clone of the GlobalStat website (www.trigger.eui.eu). This makes the website independent and more flexible in terms of customisations. The key left hand menu structure has three key 'Themes': Data Repository (A), Data Viz (B), EU in Global Governance (C). Underneath the first level of 'Themes', subthemes display the visualisation of data. Data integrated in AGGREGATOR will be open source.

AGGREGATOR functions on the one hand as (A) the central data repository of TRIGGER datasets, meaning that dataset files are downloadable from the website and that metadata is added in the website to explain the content of the datasets.

Moreover, (B) parts of the data from the datasets are represented in dynamic data visualisations to visualise certain elements and developments related to European and global governance. For these data visualisations, data from the original TRIGGER datasets are harmonised and manually integrated into the AGGREGATOR back office to be visible in the website.

Finally, (C) one part of the data visualisation summarises research results on EU actorness and effectiveness in a new type of text-heavy visualisation that is developed for AGGREGATOR.

3. Technical Development of AGGREGATOR

The technical development of TRIGGER AGGREGATOR benefits from the long-standing cooperation of the Global Governance Programme of the Robert Schuman Centre for Advanced Studies at the European University Institute and the Organisation for Economic Co-Operation and Development (OECD) within the framework of the GlobalStat database project.¹ Ever since its launch in 2015, GlobalStat has become an important resource and cooperation partner for academics and public officials. It has constantly increased and diversified its data pool and data visualisation portfolio. Through cooperation with the European Parliamentary Research Service (EPRS), it has become a knowledge partner of the European Parliament. GlobalStat cooperates with the OECD within an open data management and visualisation software development consortium that also collaborates with the European Central Bank (ECB). The outcome of this cooperation combines state-of-the-art SDMX techniques for automated data updates with new and advanced data visualisation tools and offers innovative options for the comparison of data across policy areas.

TRIGGER AGGREGATOR is based on a software clone of the well-established GlobalStat data management software. Like GlobalStat it is therefore based on the ‘Compare your country’ data dissemination platform, a software jointly developed with and managed by the OECD.² The latter includes three independent modules:

1. A user-facing data visualisation and data browsing interface which allows users to plot statistical information in various ways and to search for any given indicator. Data is presented with rich metadata information and can be combined with textual information, structured by country or any other set of entities. This tool includes some basic PHP modules, but is mainly written in JavaScript, using Webpack as package manager. All

¹ <https://globalstat.eu/>

² <https://www1.compareyourcountry.org/en>

components are open source and licensed under MIT licences or equivalent with the exception of the HighCharts data visualisation library which comes under a commercial licence.

2. A back office to manage the collections of data and metadata. The tool uses a MySQL database and PHP frameworks (Slim for route management, Eloquent as ORM mapper and Symphony as data validator) to manage server side interactions. The user-facing part uses Bootstrap modules and Webpack as package manager. All components are open source and licensed under MIT licences or equivalent.
3. A server side module to retrieve data from SDMX data warehouses and convert SDMX data into a JSON format used by the Compare your country framework.

As part of the software customisation for TRIGGER AGGREGATOR, additional modules have been added to the existing GlobalStat data management software (such as text-heavy data visualisations or the embedment of documents and other websites or data visualisations through iframe-elements).

4. Technical details on embedded data visualisations

As outlined in D3.6 "Data visualisations and dynamic presentations of data", AGGREGATOR follows a modular approach, with different approaches adapted to the types of visualisation and types of data. This chapter provides more technical details regarding the external visualisation modules embedded in the main AGGREGATOR website.

The most important example is the EurLex web application, which will be embedded in AGGREGATOR with an “iframe” link. The HTML Inline Frame element (<iframe>) allows web developers to nest one webpage into another webpage.³ This means that we can host visualisations like the EurLex web application on a different server, using different technologies than the main website (www.trigger.eui.eu) and still display it on the main AGGREGATOR website. In the case of the EurLex web application, the app is hosted on an Amazon Web Services (AWS) server (the **back end**). The **front end** of the application is programmed with the programming language Python.

4.1. Back end – Hosting via Amazon Web Services (AWS)

The EurLex web application uses the AWS service “Elastic Beanstalk”.⁴ Elastic Beanstalk is an industry leading service for deploying and scaling web applications. We are using the cloud

³ <https://developer.mozilla.org/en-US/docs/Web/HTML/Element/iframe>

⁴ <https://aws.amazon.com/elasticbeanstalk/>

D3.7 Technical Brief on AGGREGATOR

servers and services provided by AWS in Paris, France. AWS bundles several web services which are necessary for running a web application in the cloud.

- **Elastic Compute Cloud instances (EC2):** The web application needs a processor to run the calculations necessary for responding to user inputs. We are using cloud CPUs (virtual machines) to run these calculations. EC2 allows us to dynamically scale the CPUs up and down to adapt the required processing power according to user demand.
- **Simple Storage Service (S3) bucket:** The web application needs to store the datasets relevant for data visualisation. S3 provides a cloud storage service, which is used to store e.g. the EurLex dataset and make it directly available for the EC2 instance when users request data.
- **Security groups:** AWS provides an instance security group and load balancer security group to secure the HTTP requests coming from users.
- **Domain name:** A domain name (a URL/link) which users can use to access the application via their browser. This URL is currently “<http://dash-env.eba-jkz6ua3p.eu-west-3.elasticbeanstalk.com/>”. It is important to note, however, that users will not directly use this link to see the web application, but we will embed this link in the AGGREGATOR website via iframe. This will make the app visible on the AGGREGATOR website, without users having to navigate to the original (less appealing) link.
- **Other services:** Elastic Beanstalk includes several other services managed by AWS, such as an auto scaling group, to ensure a smooth functioning of the application.

These services make up the **back end** of the embedded interactive visualisation. They constitute the underlying cloud infrastructure which serves the web application to users via the web browser. Using an external service like AWS has several advantages for the TRIGGER team: (1) We do not have to set up our own dedicated IT infrastructure, which would incur significant monetary and human resource costs; (2) We can limit the resources to invest in the maintenance of the IT infrastructure; (3) We can scale the different services more flexibly, depending on user demand.

Before deploying the web application to AWS, it needs, however, to be developed locally first. The web application framework used for creating the application is the Dash/Flask framework in the programming language Python. The Python code and datasets are uploaded to AWS, which then creates the app in the cloud.⁵ For parts of this code see Annex 1.

4.2. Front end – Visualisation with Dash

The main framework for developing the web application is the open source library Dash⁶. Dash is a Python framework specifically designed for creating interactive data visualisations via web

⁵ <https://docs.aws.amazon.com/elasticbeanstalk/latest/dg/create-deploy-python-flask.html#python-flask-deploy>

⁶ <https://dash.plotly.com/>

D3.7 Technical Brief on AGGREGATOR

applications. It is itself built on top of the popular Flask framework, one of the two leading Python web development frameworks.⁷

It is important to note that web browsers do not natively understand Python code. Web browsers mainly render website based on HTML (for the skeleton of the website), CSS (to improve visuals) and JavaScript (for interactivity). Dash therefore wraps around the most important HTML, JavaScript and CSS components and makes them accessible to Python developers in pure Python. This means that the web application is written in pure Python, which is later automatically translated to HTML, CSS and JavaScript. In addition, the visualisations (line charts, bar charts) are written with Plotly⁸, an open source plotting library for Python.

This combination of open source technologies is used to create the interactive web application front end. **Annex 1.** shows parts of the Python Dash code the EurLex application. The Python code is first developed on a local computer and then deployed to the AWS back end to make it available online.

5. Content of AGGREGATOR

In its data repository function (A), the content of AGGREGATOR displays Excel and PDF files that are available for download; metadata describes the datasets. The data repository of AGGREGATOR consists of the following files (see table 1).

Table 1 - AGGREGATOR data repository

	Title	File type	TRIGGER Del #
1	Dataset on existing global governance regimes, tools and approaches	xls	D1.1
1a	Codebook for the Dataset on existing global governance regimes, tools and approaches	pdf	
2	Dataset on International Regulatory Cooperation	xls	D1.2
2a	Codebook for the Dataset on International Regulatory Cooperation	pdf	
3	Dataset on transnational private regulatory regimes	xls	D1.3
3a	Codebook for the Dataset on transnational private regulatory regimes	pdf	
5	Dataset on measuring global governance instruments	xls	D1.4
5a	Codebook for the Dataset on measuring global governance instruments	pdf	
6	Atlas of the EU's involvement in global governance	pdf	D1.6
7	Database on EU differentiated integration	xls	D2.2

⁷ <https://flask.palletsprojects.com/en/1.1.x/>

⁸ <https://plotly.com/>

D3.7 Technical Brief on AGGREGATOR

7a	Codebook for the Database on EU differentiated integration	pdf	
8	Database on EU governance arrangements per policy area, including associated impact assessments	xls	D2.3
8a	Codebook for the Database on EU governance arrangements per policy area, including associated impact assessments	pdf	
9	Database on implementation and enforcement mechanisms within EU multi-level governance	xls	D2.4
9a	Codebook for the Database on implementation and enforcement mechanisms within EU multi-level governance	pdf	
10	Working paper on public-private governance and regulatory instruments at the EU level	pdf	D2.5
11	Working paper: testing EU actorness and influence in domestic and global governance	pdf	D3.1
12	Toolkit to analyse EU actorness and influence in domestic and global governance	pdf	D3.2
13	Online atlas of the EU's involvement in global governance, i.e. software development for open data access	pdf	D3.5
14	Technical Brief on AGGREGATOR	pdf	D3.7

For (B), dynamic data visualisations, GlobalStat data visualisation options are used where appropriate (e.g. timeseries). Moreover, additional data visualisations (such as for European governance data) are embedded in the website through iframe embedment (see chapter 4).

Concerning (C), the visualisation of EU actorness and effectiveness, the TRIGGER Deep Dives on Climate Change, Data Protection, EU-Africa Relations, and Sustainable Development deliver the content of the text-heavy data visualisation websites. Data is integrated into the back office through JSON input.

6. Conclusion

This technical brief provides an overview of the different technologies used for AGGREGATOR. It provides an overview of AGGREGATOR and describes the technologies used for both the main AGGREGATOR website (chapter 3) as well as the embedded visualisations (chapter 4). See annex 1 for more technical details and a concrete code example. AGGREGATOR itself will be integrated into PERSEUS. This will bring the different components of the TRIGGER project together in one integrated user interface.

Annex 1. Eurlex app code example

Annex_1_EurLex_app_example_D3.7

July 28, 2020

1 The EurLex application - HTML, CSS, JS written in Python

The following two segments provide examples of the code for the EurLex web application (as of July 2020). Both are written in pure Python and show how HTML, CSS and JavaScript are indirectly available via the Dash web development framework. The first segment shows the initial layout of the application. The second segment shows how the layout is updated via callbacks based on user input. Please note that these code segments are only parts of the application and they are still under development.

1.1 The application layout - HTML and CSS

```
[ ]: application = dash.Dash(__name__, external_stylesheets=external_stylesheets)

app = application.server

application.layout = html.Div(style={'width': '80%', 'margin': '0 auto',
                                     #'display': 'inline-block',
                                     'alignItems': 'center', 'justifyContent': 'center',
                                     'paddingBottom': '150px', 'paddingTop': '20px',
                                     'fontSize': '12px'},
                               children=[ # master div children
                                         ## headline blocks
                                         html.H2("Interactive Analysis of 140.000+ EU laws",
                                                 style={'textAlign': 'center'}),

                                         dcc.Markdown(children=md_text_1, style={'textAlign': 'center', 'fontSize': '13px'}),

                                         html.Br(),


                                         ##### Tabs ----
                                         dcc.Tabs(vertical=False, style={'display': 'block', 'marginLeft': 'auto',
                                                                        'marginRight': 'auto', 'width': '100%'},
                                         children=[

                                         ### TAB 1
```

```

dcc.Tab(label='1. EU Legal Authority', style={'fontSize': '12px'},
       children=[

        ### div for 2 boxes within tab
        html.Div(style={'display': 'flex'}, children=[

            ### html div box 1
            html.Div(style={'display': 'inlineBlock', 'width': '40%', □
                           ↵'paddingRight': '20px'},
                     children=[

                         html.Br(),
                         dcc.Markdown("##### Analyse the EU's legal authority in any□
                           ↵topic",
                                     style=css_centre
                         ),
                         dcc.Markdown(children=md_indiv_analysis,
                                     style={**css_centre, 'textAlign': 'justify', □
                           ↵'fontSize': '12px'}
                         ),
                     ],

            ## select topic keywords
            html.Br(),
            dcc.Markdown(children="**Define a policy area with relevant□
                           ↵keywords**",
                         style={**css_centre, 'fontSize': '14px'}),
            html.Div(style={'display': 'flex'}, children=[

                dcc.Markdown(children="Sub-topic keywords",
                             style={'display': 'inlineBlock', 'width': □
                               ↵'110%'})
                ),
                dcc.Markdown(children="Policy area keywords",
                             style={'display': 'inlineBlock', 'width': □
                               ↵'90%', 'paddingLeft': "6%"}),
                ],
            ),
            html.Div(style={'display': 'flex'}, children=[

                dcc.Dropdown(
                    id='dropdown-indiv-ev',
                    options=ev_dic_lst, multi=True,
                    value=["data protection", "protection of privacy", □
                           ↵"personal data"],
                    style={'display': 'inlineBlock', 'width': '110%'})
                ),
                dcc.Dropdown(
                    id='dropdown-indiv-sm',
                    style={...}
                )
            ])
        ])
    ]
)

```

```

        options=sm_dic_lst, multi=True,
        #value=[],
        style={'display': 'inlineBlock', 'width': '90%', □
→'paddingLeft': "6%"})
),
]),
dcc.Markdown(children="Select the time period you are □
→interested in",
            style={'paddingTop': '10px'}),
dcc.Dropdown(id='start-date-indiv', placeholder='1980',
            options=[{'label':x, 'value': x} for x in range(1952, □
→2020)],
            #value='1980',
            style={**css_centre, 'width': '130px', 'display': □
→'inline-block'})
),
dcc.Dropdown(id='end-date-indiv', placeholder='2019',
            options=[{'label':x, 'value': x} for x in range(1952, □
→2020)],
            #value='2019',
            style={**css_centre, 'width': '130px', 'display': □
→'inline-block'})
),
html.Br(),
html.Div(style={'display': 'flex'}, children=[
    html.Button(id='submit-fig-indiv', n_clicks=0, □
→children='Create figure',
            style={'display': 'inline-block', □
→'fontSize': '9px'},
            ),
    dcc.Checklist(id='checklist-average-authority',
                  options=[{'label': 'Include authority □
→score for average sub-topic',
                           'value': 'ev'},
                           {'label': 'Include authority □
→score for average policy area',
                           'value': 'sm'}],
                  value=['ev'],
                  # labelStyle={'display': 'inline-block'},
                  style={'display': 'inline-block', □
→'paddingLeft': '20px'})
),
])
),

### compare legal authority analysis
html.Br(),

```

```

        html.Br(),
        dcc.Markdown(children="**Compare with a different topic**",
                     style={**css_centre, 'fontSize': '14px'}),
        html.Div(style={'display': 'flex'}, children=[
            dcc.Markdown(children="Sub-topic keywords",
                         style={'display': 'inlineBlock', 'width': 'auto',
                                'paddingRight': '6%'},
                         ),
            dcc.Markdown(children="Policy area keywords",
                         style={'display': 'inlineBlock', 'width': 'auto',
                                'paddingLeft': '90%'},
                         ),
        ]),
        html.Div(style={'display': 'flex'}, children=[
            dcc.Dropdown(
                id='dropdown-indiv-compare-ev',
                options=ev_dic_lst, multi=True,
                #value=["data protection", "protection of
                #privacy", "personal data"],
                style={'display': 'inlineBlock', 'width': 'auto',
                       'paddingLeft': '110%'}
                ),
            dcc.Dropdown(
                id='dropdown-indiv-compare-sm',
                options=sm_dic_lst, multi=True,
                # value=[],
                style={'display': 'inlineBlock', 'width': 'auto',
                       'paddingLeft': '90%', 'paddingRight': '6%'}
                ),
        ]),
        html.Button(id='submit-fig-indiv-compare', n_clicks=0,
                    children='Add custom topic to figure',
                    style={'fontSize': '9px'}),
    ]),
    # end of first box in tab

    ### html div box 2
    #figures
    html.Div(id='fig-indiv-line',
             style={**css_centre, 'display': 'inlineBlock', 'width': 'auto',
                    'paddingTop': '40px'},
             children=dcc.Graph(figure=fig_1)
    )

```

```

]), # end of second box in tab

#### Table
html.Div(id="indiv-table-div",
    children=[
        html.Br(),
        dcc.Markdown(style={'textAlign': 'center'},
            children=["##### **Raw Data Table**"]),
        html.Div(style={'display': 'flex'}, children=[
            html.Div(style={**css_centre, 'display': 'inlineBlock', 'width':
                '90%'},,
                children=[

                    dcc.Markdown(id='table-text', children=md_table,
                        style={'fontSize': '12px'})
                ]),
            html.Div(style={**css_centre, 'display': 'inlineBlock', 'width':
                '10%',,
                    'paddingLeft': '10px'},
                children=[

                    html.Button(id='button-download-indiv', n_clicks=0, style={
                        'fontSize': '9px'},
                        children=[

                            html.A("Download Raw data", id="download-link",
                                href="/dash/DownloadDataIndiv?ev-keywords=['data\u2019protection', 'protection of privacy', 'personal\u2019data']&sm-keywords=None&start-date=1980&end-date=2020")
                        ]),
                ])
            ],
        html.Br(),
    ]
)

# DataTable
dash_table.DataTable(
    id='table-acts-topic',
    data=df_topic_y_filt_1.to_dict('records'),
    columns=[{"name": i, "id": i} for i in df_topic_y_filt_1.
    columns],
    page_size=10,

    #style_cell={'minWidth': 95, 'width': 95, 'maxWidth': 1000}
    style_table={'overflowX': 'auto'}, # horizontal scroling
    style_cell={'textAlign': 'left', # all cells
                'minWidth': '180px', 'width': '180px', 'maxWidth': '300px',
                'fontSize': '10px'},

```

```

        style_header={'fontWeight': 'bold', 'backgroundColor': 'white', 'color': 'black', 'border': '1px solid black', 'padding': '5px', 'margin': '5px 0'},
        style_data={'whiteSpace': 'normal', # only data
                    #'height': 'auto', 'lineHeight': '15px',
                    'overflow': 'hidden', 'textOverflow': 'ellipsis',
                    'maxWidth': 0
                },
            ## constrain hight of cells
            css=[{'selector': '.dash-spreadsheet td div',
                  'rule': '''
                        line-height: 15px;
                        max-height: 100px; min-height: 30px; height: 70px;
                        display: block;
                        overflow-y: hidden;
                        ...
                      '''],
            tooltip_data=[{
                column: {'value': str(value), 'type': 'markdown'}
                for column, value in row.items()
                } for row in df_topic_y_filt_1.to_dict('rows')
            ],
            tooltip_duration=None,
            style_data_conditional=[{
                'if': {'row_index': 'odd'},
                'backgroundColor': 'rgb(248, 248, 248)'
            }],
            ),
            ]),
        ## bar charts parked at end of first tab. unclear if delete
        html.Br(),
        html.Br(),
    ]), ## End of first tab

    # other tabs
    dcc.Tab(label='2. Actors in EU law', style={'fontSize': '12px'},
            children=[
                html.H4("Content to be developed"))),
    dcc.Tab(label='3. EU Differentiated Integration', style={'fontSize': '12px'},

```

```

        children=[  

            html.H4("Content to be developed")])  

    ]),  

])

```

1.2 Callback example - adding interactivity with callbacks

```
[ ]: @application.callback(  

    [Output('fig-indiv-line', 'children'),  

     Output('table-acts-topic', 'data'),  

     Output('table-acts-topic', 'columns'),  

     Output('table-acts-topic', 'tooltip_data'),  

     Output('table-text', 'children'),  

     Output('download-link', 'href')],  

    [Input('submit-fig-indiv', 'n_clicks'),  

     Input('submit-fig-indiv-compare', 'n_clicks')],  

    [State('dropdown-indiv-ev', 'value'),  

     State('dropdown-indiv-sm', 'value'),  

     State('fig-indiv-line', 'children'),  

     State('start-date-indiv', 'value'),  

     State('end-date-indiv', 'value'),  

     State('dropdown-indiv-compare-ev', 'value'),  

     State('dropdown-indiv-compare-sm', 'value'),  

     State('checkbox-average-authority', 'value'),  

    ]  

)
def update_topic_indiv(Clicks_submit, Clicks_compare, Dropdown_ev, Dropdown_sm,  

                      Fig_main, Start_date, End_date,  

                      Dropdown_compare_ev, Dropdown_compare_sm,  

                      Checklist_average_authority):  

  

    ctx = dash.callback_context # to access more than click count from buttons  

    # https://dash.plotly.com/advanced-callbacks  

  

    if ctx.triggered[0]['value'] == 0:  

        return dash.no_update, dash.no_update, dash.no_update, dash.no_update,  

        dash.no_update, dash.no_update, \  

            #dash.no_update, dash.no_update  

    if (Dropdown_ev is None or len(Dropdown_ev) == 0) and (Dropdown_sm is None  

    or len(Dropdown_sm) == 0):  

        return dash.no_update, dash.no_update, dash.no_update, dash.no_update,  

        dash.no_update, dash.no_update, \  

            #dash.no_update, dash.no_update  

  

    # date range
```

```

if Start_date is None:
    Start_date = 1980
if End_date is None:
    End_date = 2020

# which of the two buttons was clicked?:
if ctx.triggered[0]["prop_id"] == "submit-fig-indiv.n_clicks":
    if Dropdown_ev is None or len(Dropdown_ev) == 0:
        dropdown_input_ev = "NoKeywordABC"
    elif len(Dropdown_ev) > 1:
        step1 = ";"|.join(Dropdown_ev) + ";" # to match keyword segment in
→string (";" is the separator between keyword segments)
        step2 = "$|".join(Dropdown_ev) + "$" # in case keyword is end of
→string
        dropdown_input_ev = step1 + step2
    else:
        dropdown_input_ev = Dropdown_ev[0] + ";"| + Dropdown_ev[0] + "$"
# format sm as regex
if Dropdown_sm is None or len(Dropdown_sm) == 0:
    dropdown_input_sm = "NoKeywordABC"
elif len(Dropdown_sm) > 1:
    #dropdown_input = ";"|.join(Dropdown)
    step1 = ";"|.join(Dropdown_sm) + ";" # to match keyword segment in
→string (";" is the separator between keyword segments)
    step2 = "$|".join(Dropdown_sm) + "$" # in case keyword is end of
→string
    dropdown_input_sm = step1 + step2
else:
    dropdown_input_sm = Dropdown_sm[0] + ";"| + Dropdown_sm[0] + "$"
elif ctx.triggered[0]["prop_id"] == "submit-fig-indiv-compare.n_clicks":
    if Dropdown_compare_ev is None or len(Dropdown_compare_ev) == 0:
        dropdown_input_ev = "NoKeywordABC"
    elif len(Dropdown_compare_ev) > 1:
        #dropdown_input = ";"|.join(Dropdown)
        step1 = ";"|.join(Dropdown_compare_ev) + ";" # to match keyword
→segment in string (";" is the separator between keyword segments)
        step2 = "$|".join(Dropdown_compare_ev) + "$" # in case keyword is
→end of string
        dropdown_input_ev = step1 + step2
    else:
        dropdown_input_ev = Dropdown_compare_ev[0] + ";"| +
→Dropdown_compare_ev[0] + "$"
# format sm as regex
if Dropdown_compare_sm is None or len(Dropdown_compare_sm) == 0:
    dropdown_input_sm = "NoKeywordABC"

```

```

    elif len(Dropdown_compare_sm) > 1:
        #dropdown_input = ";" / ".join(Dropdown)
        step1 = ";" | ".join(Dropdown_compare_sm) + ";" | " # to match keyword
        ↵segment in string (";" is the separator between keyword segments)
        step2 = "$| ".join(Dropdown_compare_sm) + "$" # in case keyword is
        ↵end of string
        dropdown_input_sm = step1 + step2
    else:
        dropdown_input_sm = Dropdown_compare_sm[0] + ";" | "
        ↵Dropdown_compare_sm[0] + "$"
    else:
        dropdown_input_ev = "NoKeywordABC" # Dropdown_ev
        dropdown_input_sm = "NoKeywordABC" # Dropdown_sm

    # get topic df
    df_topic_all_update = df_topic_filter(Re_ev=dropdown_input_ev,
    ↵Re_sm=dropdown_input_sm, Start_Date=Start_date, End_Date=End_date,
                                df_acts_reg=df_acts_reg,
    ↵df_acts_dir=df_acts_dir, df_acts_dec=df_acts_dec,
                                df_com=df_com,
    ↵df_internat=df_internat)[1] # df_topic_y_filt_update excluded because
    ↵generated in next function
    # calculate authority score
    df_topic_y_filt_update, authority_score_y =
    ↵legal_authority_score(df_topic_all_update, Start_date=Start_date,
    ↵End_date=End_date)

    # if submit-fg-indiv-compare: return updated fig
    if ctx.triggered[0]["prop_id"] == "submit-fg-indiv-compare.n_clicks":
        # update existing fig
        fig_authority_update =
    ↵plot_authority_update(Fig_main['props']['figure'], authority_score_y,
                                Start_date=Start_date,
    ↵End_date=End_date)
        return dcc.Graph(figure=fig_authority_update), dash.no_update, dash.
    ↵no_update, dash.no_update, \
        dash.no_update, dash.no_update, #dash.no_update, dash.
    ↵no_update

    ### following only if submit-fg-indiv was clicked ---
    # create legal authority line chart
    fig_authority_update = plot_authority(authority_score_y,
    ↵Start_date=Start_date, End_date=End_date)

    # add average authority to figure

```

```

mean_sm_y_update = mean_sm_y[(mean_sm_y.index >= Start_date) & (mean_sm_y.
↪index <= End_date)]
mean_ev_y_update = mean_ev_y[(mean_ev_y.index >= Start_date) & (mean_ev_y.
↪index <= End_date)]
if "sm" in Checklist_average_authority:
    fig_authority_update.add_trace(go.Scatter( # modify fig object
        x=mean_sm_y_update.index,
        y=mean_sm_y_update.iloc[:, 0], # Df_score["Legal_authority_score"],
        name="Average policy area",
        mode='lines+markers',
        line_shape='spline' # smoothes line
    )
)
if "ev" in Checklist_average_authority:
    fig_authority_update.add_trace(go.Scatter( # modify fig object
        x=mean_ev_y_update.index,
        y=mean_ev_y_update.iloc[:, 0], # Df_score["Legal_authority_score"],
        name="Average sub-topic",
        mode='lines+markers',
        line_shape='spline' # smoothes line
    )
)

### table update
topics_strings = ''
if Dropdown_ev is None or len(Dropdown_ev) == 0:
    next
elif len(Dropdown_ev) > 0:
    topics_strings = topics_strings + ' ' + ', '.join(Dropdown_ev) + ' '
elif len(Dropdown_ev) == 0:
    topics_strings = topics_strings + ' ' + Dropdown_ev[0] + ' '
if (Dropdown_ev is not None and len(Dropdown_ev) > 0) and (Dropdown_sm is
↪not None and len(Dropdown_sm) > 0):
    topics_strings = topics_strings + ', '
if Dropdown_sm is None or len(Dropdown_sm) == 0:
    next
elif len(Dropdown_sm) > 0:
    topics_strings = topics_strings + ' ' + ', '.join(Dropdown_sm) + ' '
elif len(Dropdown_sm) == 0:
    topics_strings = topics_strings + ' ' + Dropdown_sm[0] + ' '

md_table_update = f'''
Your search for legal acts related to {topics_strings} found
↪{len(df_topic_y_filt_update)} documents:
{len(df_topic_y_filt_update[df_topic_y_filt_update['Act_type'].str.
↪contains("(?i)regulation")])} Regulations,

```

```

{len(df_topic_y_filt_update[df_topic_y_filt_update['Act_type'].str.
↪contains("(?i)directive")])} Directives,
{len(df_topic_y_filt_update[df_topic_y_filt_update['Act_type'].str.
↪contains("(?i)decision")])} Decision,
{len(df_topic_y_filt_update[df_topic_y_filt_update['Act_type'].str.
↪contains("(?i)international agreement")])} International Agreements,
{len(df_topic_y_filt_update[df_topic_y_filt_update['Act_type'].str.
↪contains("(?i)communication")])} Communications.

The table below displays the underlying raw data related to the topic. For
↪details on individual documents, we recommend
visiting the official Eur-lex.eu website via the Eurlex_link.
You can download the raw data via the download button to the right.
'''

# cut df_topic_y_filt_update if too long, otherwise browser renders very
↪slowly (e.g. for trade policy with 27k~ laws)
if len(df_topic_y_filt_update) > 2000:
    df_topic_y_filt_update = df_topic_y_filt_update[:2000]

table_data = df_topic_y_filt_update.to_dict('records')
table_columns = [{"name": i, "id": i} for i in df_topic_y_filt_update.
↪columns]
# tool tip causes error for some reason if > 2000:
if len(df_topic_y_filt_update) < 2000:
    tooltip_data = [{column: {'value': str(value), 'type': 'markdown'} for
↪column, value in row.items()
} for row in df_topic_y_filt_update.to_dict('rows')]
else:
    tooltip_data = [None]

## this updates the download link and stores the parameters for the
↪computation in the link
# the download_csv() function of the server.route can then take these
↪values from the URL
# and use them for repeating the computation
if ctx.triggered[0]['value'] != 0:
    download_url = f'/dash/DownloadDataIndiv?
↪ev-keywords={Dropdown_ev}&sm-keywords={Dropdown_sm}&start-date={Start_date}&end-date={End_d

return dcc.Graph.figure=fig_authority_update), \
    table_data, table_columns, tooltip_data, md_table_update, \
↪download_url #dcc.Graph.figure=fig_bar_ev_update), dcc.
↪Graph.figure=fig_bar_sm_update),

```



Technische Universität München



POLITECNICO
MILANO 1863



ევროპული კულტურული
საქართველოს ინიციატიური
განვითარებისთვის



INSTITUTION EURASIAN INSTITUTE
OF INTERNATIONAL RELATIONS



This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement n° 822735. This document reflects only the author's view and the Commission is not responsible for any use that may be made of the information it contains.