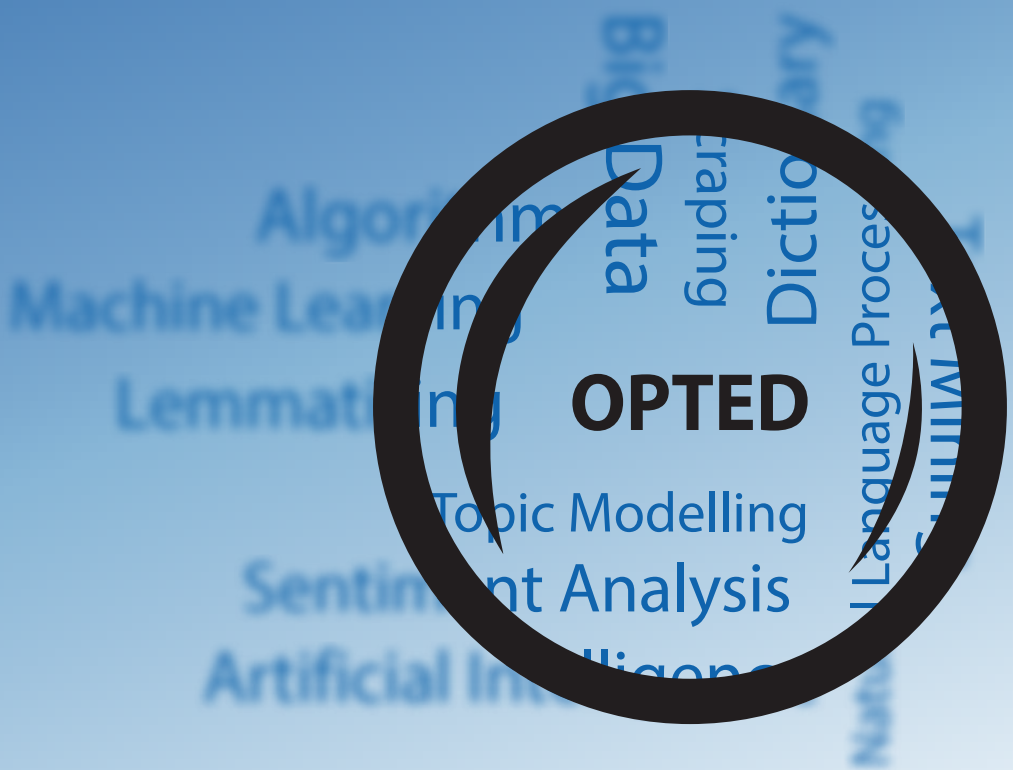
OPTED

Non-consumptive research toolkit

Deliverable 7.5

Wouter van Atteveldt, Johannes B. Gruber, and Kasper Welbers

Vrije Universiteit Amsterdam



Disclaimer

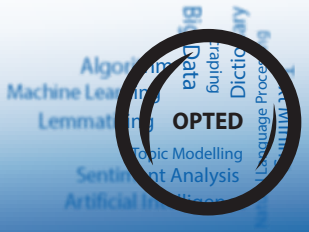
This project has received funding from the European Union's Horizon 2020 research & innovation programme under grant agreement No 951832. The document reflects only the authors' views. The European Union is not liable for any use that may be made of the information contained herein.

Dissemination level

Public

Type

Report



OPTED
Observatory for Political Texts in European Democracies:
A European research infrastructure

Non-consumptive research toolkit

Deliverable 7.5

Authors: Wouter van Atteveldt, Johannes B. Gruber, and Kasper Welbers

Vrije Universiteit Amsterdam

Due date: May 2023

Contents

Executive Summary

The overall objective of WP7 is to establish routines and protocols for varieties of standardizations of pre-processing, pending on source and purpose of usage of text. This work package focuses on assessing and providing prototypes of open science and open data structures in terms of data storage.

D7.5 extends the capabilities of AmCAT 4.0 with functions for non-consumptive research – that is research that can be performed on a text data collection computationally, while access to the original data can not be granted because of copyright, privacy or other concerns. AmCAT 4.0 enables and facilitates the access and processing of data that cannot be shared openly, by keeping data owners in direct control of the data, and by employing trusted connections and role-based access control methods. To this end, D7.5 introduces re-designed access control and guest access to AmCAT 4.0, as well as adding a flexible framework of pre-processing actions. This framework, which we called actioncat, can be used for pre-processing actions before granting the (public) access to the processed versions of text, from which the original content can not be reproduced. This way, users without access can still perform a wide range of analyses, without having to grant them access to the original data.

This deliverable consists of the AmCAT 4.0 codebase and the codebase of the extension actioncat, published via the OPTED website and publicly accessible on the GitHub repositories.

1 Non-Consumptive Research in `amcat4` and actioncat

1.1 Access Control in Amcat4 Explained

`amcat4` provides fine-grained access control, which enables administrators of an instance to share data in exactly the way they want and control what members of their team and outsiders can do on an index. This section explains how access control works in `amcat4` and highlights which features and settings provide infrastructure for non-consumptive research.

We provide two sets of roles: one that is configred globally per user on an `amcat4` instance and one set that can be used per index. In combination the three global (reader, writer and admin) and index roles (none, metareader, reader, writer and admin) offer twelve different role levels that are shown in the table below:

Relevant for non-consumptive research are the metareader levels. Users with this index role can perform analysis on the data via the AmCAT dashboard or the API, but do not have access to the text data.

Additionally, we can control the default role users have on a given index (see level 0 in the table). We call this the guest role of an index. This way, it is possible, for example, to have one index where everyone can see the metadata while other indexes on the AmCAT instance are hidden.

Users' global and index role and the guest roles of indexes can be controlled via the dashboard or the API. We show this illustrativly in the example below.

However, first we need to highlitght another setting that can modify access to data in `amcat4`: the authentification mode. The authentification mode of an `amcat4` instance

Table 1: Access roles in amcat4

Role Level	Global Role	Index Role In Index A	Description
0	-	Guest	If a user does not have an explicit role on an index, the guest role (if any) is used. An unauthorized user can still get guest roles, so they can see any indices with a guest role. This is not a specific level, but it serves as a fallback for undefined index roles.
1	Reader	None	Can see which indices exist, but cannot access them. Index A is invisible to the user.
2*	Reader	Metareader	Can see which indices exist. Can read all properties, do queries, etc. in Index A, but cannot read the 'text' attribute.
3	Reader	Reader	Can see which indices exist. Can read all properties, do queries, etc. in Index A, but cannot make changes.
4	Reader	Writer	Can see which indices exist. Can add/delete documents, add/delete users (up to their own level), and make other changes (but not delete) in Index A.
5	Reader	Admin	Can see which indices exist, can add/update/delete documents and users (up to their own level) in Index A and delete Index A itself.
6	Writer	None	Can create new indexes and users (with at most their own global role). Index A is invisible to the user.
7*	Writer	Metareader	Can create new indexes and users (with at most their own global role). Can read all properties, do queries, etc. in Index A, but cannot read the 'text' attribute.
8	Writer	Reader	Can create new projects and users (with at most their own global role). Can read all properties, do queries, etc. in Index A, but cannot make changes.
9	Writer	Writer	Can create new projects and users (with at most their own global role). Can add/delete documents, add/delete users (up to their own level), and make other changes in Index A, but can't delete Index A.
10	Writer	Admin	Can create new projects and users (with at most their own global role). Can add/update/delete documents and users (up to their own level) in Index A and delete Index A itself.
11	Admin	Admin	Can delete projects and assign themselves a role on any index role. Can do whatever they want, including deleting the index.

* Relevant for non-consumptive research

controls who has access to the data in the first place. Unlike the roles, this setting can only be accessed by administrators through the command line on the machine where the instance is hosted. If you follow our recommended way of installation (see the online manual), `amcat4` will run in a Docker container. When invoked with the command `docker exec -it amcat4 amcat4 config`, an interactive configuration menu will guide the user through various settings. The authentication settings will show the follwing explanation:

Listing 1 Authentication modes in amcat4

```
auth: Do we require authorization?
  Possible choices:
  - no_auth: everyone (that can reach the server) can do anything they want
  - allow_guests: everyone can use the server, dependent on index-level guest_role authorization settings
  - allow_authenticated_guests: everyone can use the server, if they have a valid middlecat login, and dependent on index-level guest_role authorization settings
  - authorized_users_only: only people with a valid middlecat login and an explicit server role can use the server

The current value for auth is AuthOptions.no_auth.
Enter a new value, press [enter] to leave unchanged, or press [control+c] to abort:
```

The explanations of the different levels should hopefully be clear, except the term **middlecat login**. For authentication modes beyond allow_guests, people whom you want to grant some kind of access need to log in. To make this more secure, we wrote our own authentication provider called **middlecat**. It enables the administrator to set up authentication via different identity providers like Google or GitHub, with a fallback solution to let users log in via a one-time email link (if they want to log in again, they need to request a new link). We host a middlecat instace at https://middlecat.up.railway.app, but as the software is open source, everone can set up their own instance and negotiate with different identity providers to make other authentication options available (e.g., through their university or company).

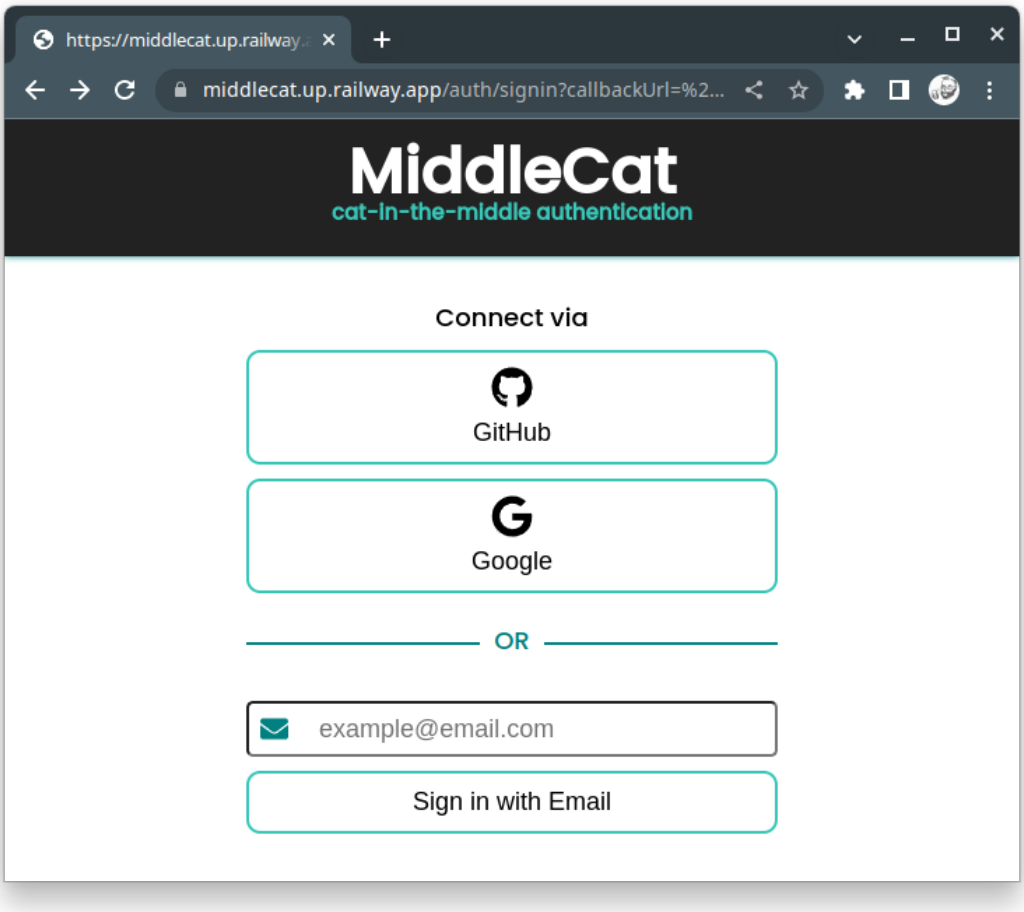


Figure 1: Middlecat login screen

In cases where non-consuptive research should be made possible, all modes except *no_auth* can be used. The indexes can then be configured to let users have a metareader role explicitly or via the guest role of an index, as explained above.

1.2 Non-Consumptive Research via the AmCAT Dashboard: an Example

As Illustration of the access control, we can use an instance where the authentication mode is set to *allow_guests* and where we added a test index. To reproduce this setup, one could spin up an instance of the AmCAT suite using Docker and Docker Compose (you can find a more detailed explanations in the AmCAT manual):

We can then change the guest role of the test index through the web dashboard at http://localhost/ (if you hosted the instance locally on your computer):

1. Log into the dashboard

Listing 2 Creating an AmCAT instance through Docker

```
# download our docker compose file with curl or manually
curl -O https://raw.githubusercontent.com/ccs-amsterdam/actio
→ ncat/main/actions/dfm/docker-compose.yml
# run docker compose to download and start the AmCAT applicat
→ ions
docker-compose up --pull="missing" -d
# create a test index to use in this example
docker exec -it amcat4 amcat4 create-test-index
# configure the instance to run in allow_guests (make sure
→ to also set your email address as admin email, or you are
→ locked out)
docker exec -it amcat4 amcat4 config
```

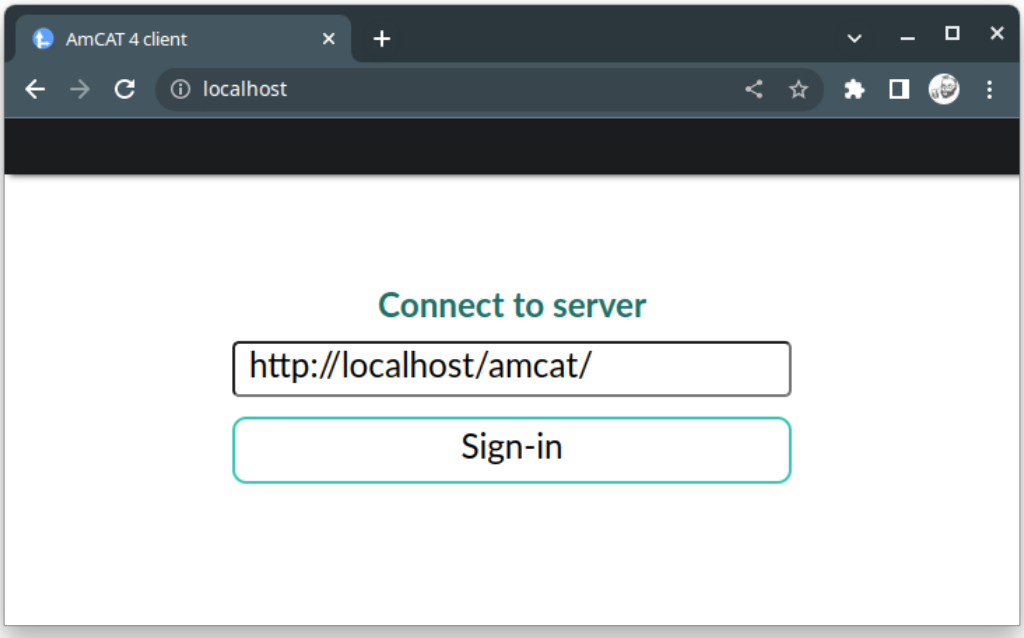


Figure 2: Web dashboard of AmCAT

2. Select the *state_of_the_union* index and go to settings:

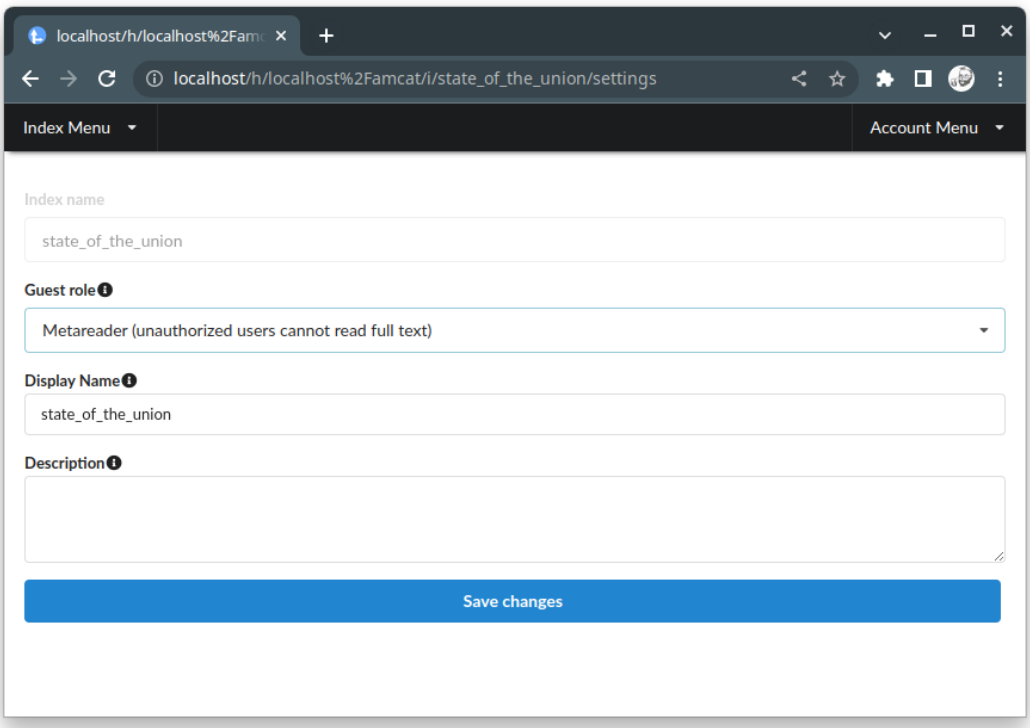


Figure 3: Change guest role of index state_of_the_union

After this, both users with and without access to the text data are still able to perform simple frequency analyses using text queries:

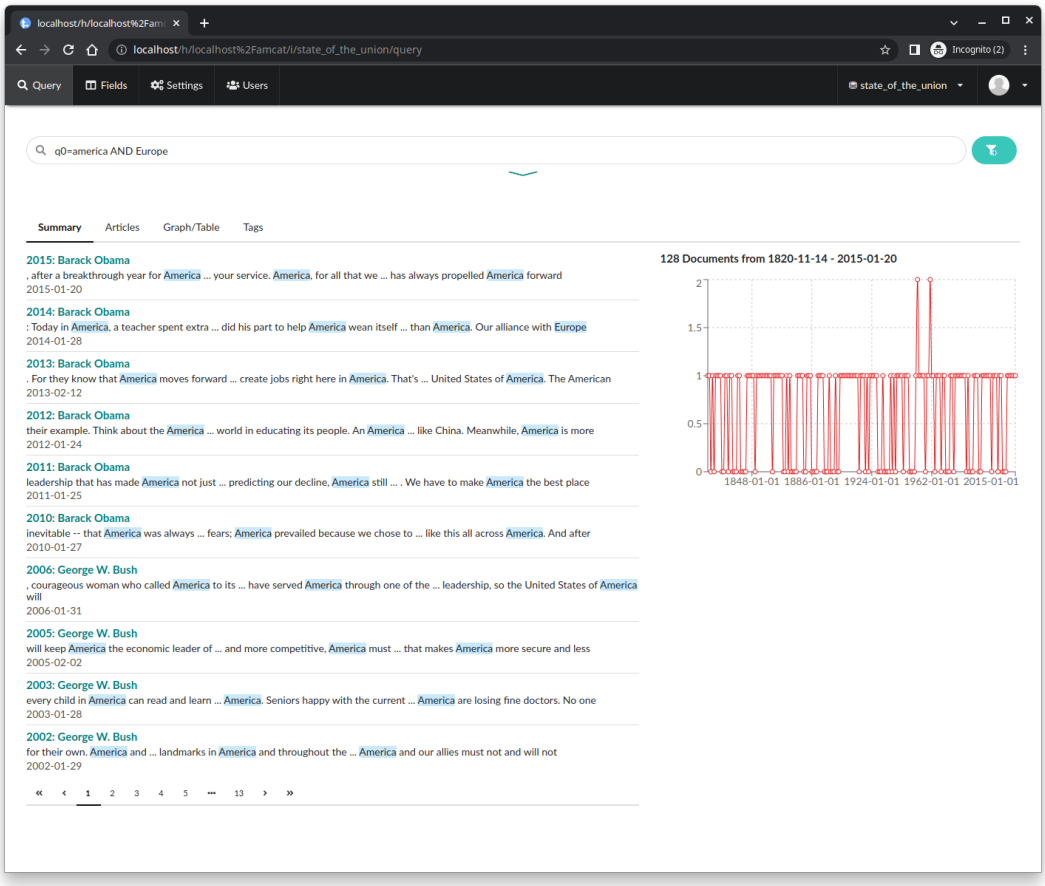


Figure 4: Frequency analysis

Users can also compare document groups based on meta fields:

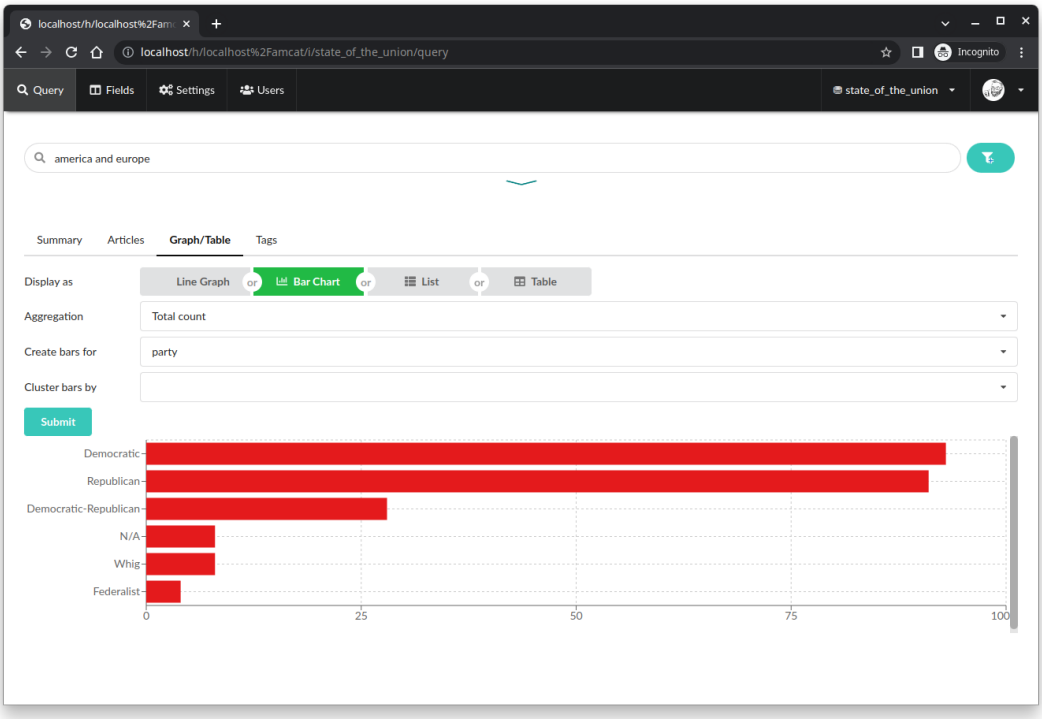


Figure 5: Comparison based on party in the state_of_the_union index

Only when users try to see more of a text than a short snippet do the index roles matter:

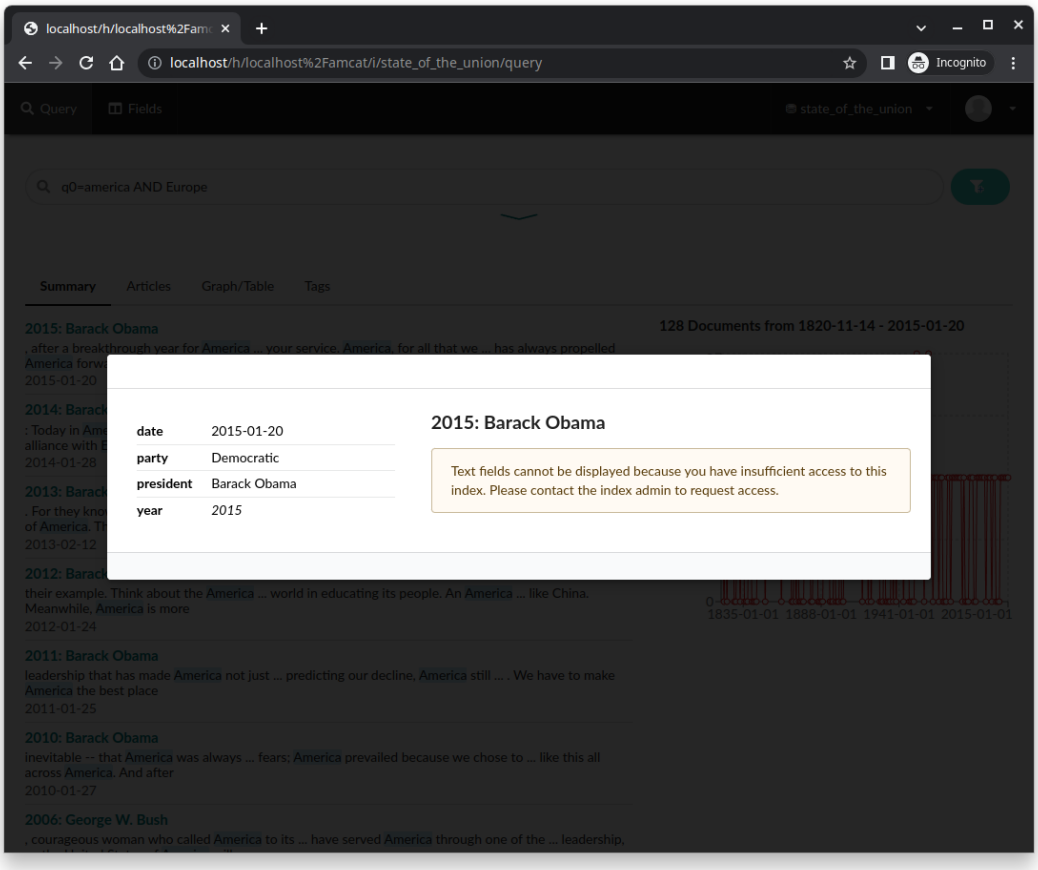


Figure 6: User with no role

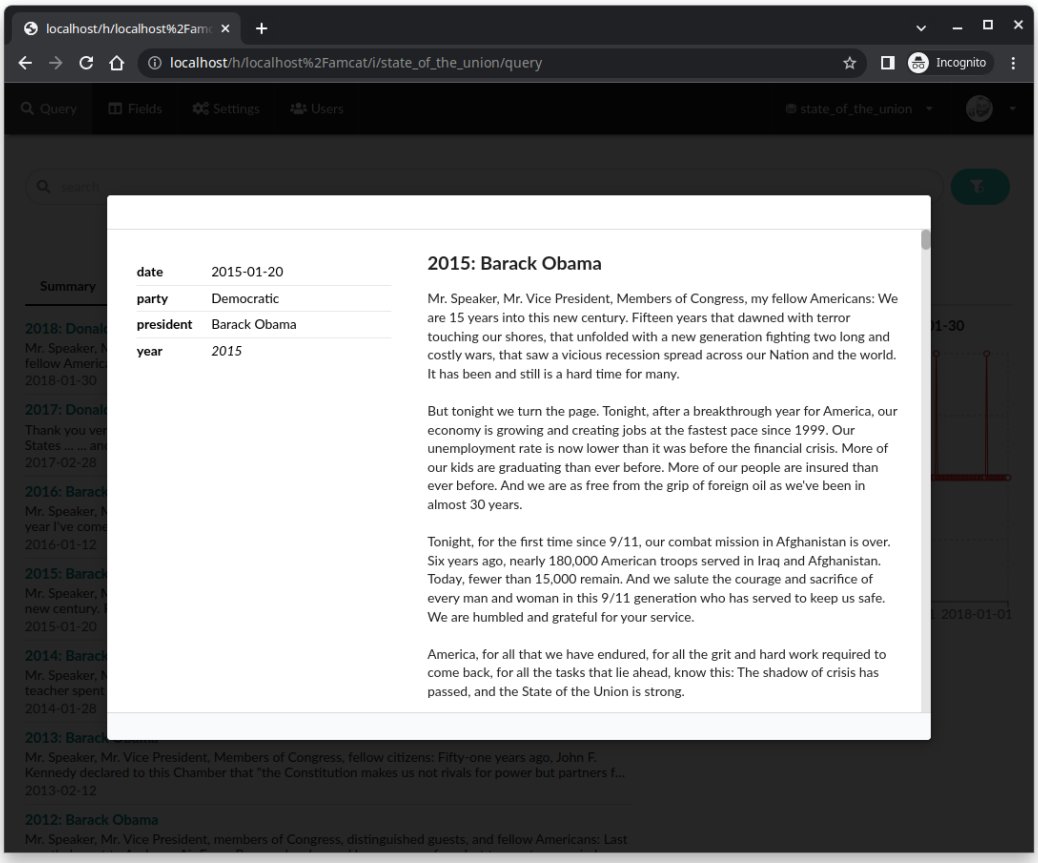


Figure 7: User with reader role

As explained above, we could have accomplished the same by assiging a specific user either role level 2 or 7.

Through the fine-grained access control and the dashboard, AmCAT offers ways to query, visualise and analyse text data without giving users access to the underlying data. This allows researchers to make datasets available for non-consuptive research where it is not possible to publicly share the full data due to copyright, privacy, or other concerns.

2 Packaged Non-Consumptive Analysis Workflows

While `amcat4` offers rudimentary analysis methods for users with metareader access like counting and cross-tabulating documents using queries, we also developed and addon framework to AmCAT that makes it possible to run entire analysis workflows in a non-consuptive way. We call this framework `actioncat` The basic idea we implemented is to leverage the open source Docker infrastrucre, which we also employ to make the AmCAT suite of packages available, to let users create specialised workflows in order to perform analyses on data they do not have access to.

We offer two example *actions* (which is what we call predefined workflows that are

packaged in a Docker image/container), one in `R`, one in `Python`:

- The `R` action adds a tidy document-feature representation field to an index
- The `Python` action adds a document embeddings field to an index

Both of these actions are destructive preprocessing workflows in the sense that the original text cannot be reconstructed from the new field. This makes these actions well suited for indexes where the full text can not be shared because of copyright, privacy or other concerns. Using AmCAT's fine grained access control features, the full text can be hidden from users without specific permissions, but the preprocessed data can still be shared with a wider audience. Users can imitate these examples to create their own workflows and send them to administrators of an `amcat4` instance. After approval, the administrator can then run a workflow using just two commands: one to download and one to run the action. Compared to sending just `R` or `Python` files for processing, this approach has the advantage that the action will have all the right dependencies already and perform the action exactly as on the user's machine (therby standardizing the process to a certain degree and making the admins life a little easier).

2.1 Non-Consumptive Research via `actioncat`: an Example

To illustrate the basic steps from the administrators perspective, we use one of the example workflows in the repository here: https://github.com/ccs-amsterdam/actioncat.

First, we spin up an instance of the AmCAT suite using Docker and Docker Compose as above:

Listing 3 Creating an AmCAT instance through Docker

```
# download our docker compose file with curl or manually
curl -O https://raw.githubusercontent.com/ccs-amsterdam/actio
→ ncat/main/actions/dfm/docker-compose.yml
# run docker compose to download and start the AmCAT applicat
→ ions
docker-compose up --pull="missing" -d
# create a test index to use in this example
docker exec -it amcat4 amcat4 create-test-index
# configure the instance to run in allow_guests (make sure
→ to also set your email address as admin email, or you are
→ locked out)
docker exec -it amcat4 amcat4 config
```

Administrators can then use actions with the same basic approach:

In case of a user defined action, only the download link will be different. The action will run until it has added a tidy document-feature representation to all texts in the test index. You can check this via the web interface at http://localhost/:

Listing 4 Running the predefined dfm action

```
# download the action file with curl or manually
curl -O https://raw.githubusercontent.com/ccs-amsterdam/actio
→ ncat/main/actions/dfm/docker-compose.yml
# run the action
docker-compose up --pull="missing" -d
```

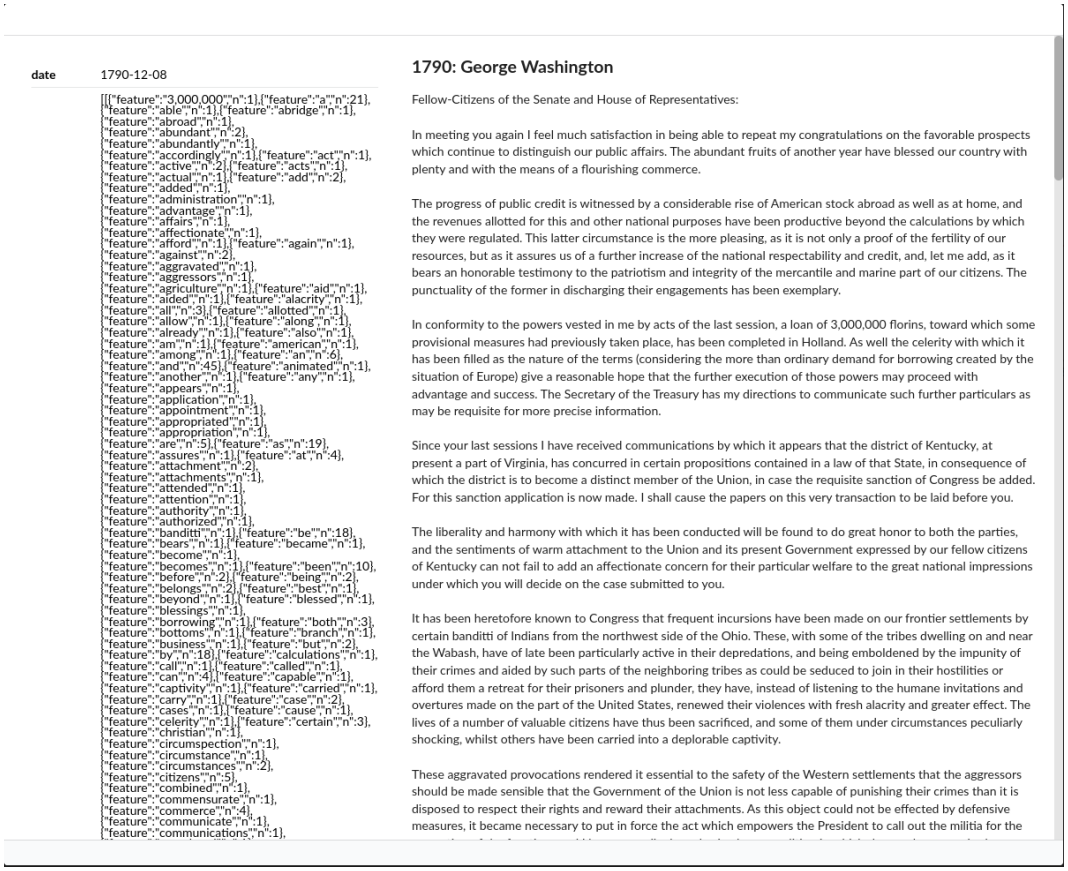


Figure 8: Example text preview including dfm

or using the `amcat4r` package:

Listing 5 Querying the new field through the API

```
if (!requireNamespace("amcat4r", quietly = TRUE))
   remotes::install_github("ccs-amsterdam/amcat4r")
library(amcat4r)
amcat_login("http://localhost/amcat")
sotu_dfm <- query_documents(
   index = "state_of_the_union",
   queries = NULL,
   fields = c(".id", "dfm"
))
sotu_dfm
```

```
# A tibble: 232 × 2
   .id      dfm
   <id_col> <list>
 1 9d8…0d0  <list [3,370]>
 2 846…068  <list [2,176]>
 3 2b6…aa5  <list [2,895]>
 4 4f3…8bf  <list [3,172]>
 5 c36…4b0  <list [3,739]>
 6 5a2…8ba  <list [3,745]>
 7 8b0…5f2  <list [3,927]>
 8 484…893  <list [3,308]>
 9 a3a…a70  <list [2,554]>
10 57c…840  <list [1,729]>
# ℹ 222 more rows
```

By changing the `docker-compose.yml`, it is possible to control which index the action is applied on, the name of the text field and the name of the new dfm field by changing the environment variables:

```
version: "3.8"
services:
  action-dfm:
    image: ccsamsterdam/amcat-action-dfm:4.0.13
    build: .
    network_mode: "host"
   environment: # behaviour of the R script is controlled through these variable
      - amcat4_host=http://localhost/amcat
      - index=state_of_the_union
      - queries=NULL
      - text_field=text
      - dfm_field=dfm
```

```
  # for authentication, this container needs access to the httr2 cache director
   # You can find it with `rappdirs::user_cache_dir("httr2")`
    # volumes:
  #   - ~/.cache/httr2:/root/.cache/httr2 # [local path]:[container path]
```

So far, we ran this on an instance without authentication. If we turn on authentication, we need to also give the container access to a valid token. This can be done by giving the action access to a valid token file. To create a token file, first log into the instance:

Listing 6 Authenticating in amcat4r

```
amcat_login("http://localhost/amcat", cache = 1L)
```

This prompt the user to log into the AmCAT instance. When `cache = 1L` is selected, a token file is written to the local computer. One can find it by following the path returned by:

Listing 7 Finding the local token directory

```
#| eval: true
rappdirs::user_cache_dir("httr2")
```

Now it is possible to link this directory to the Docker container by changing the commented out lines in `docker-compose.yml` to:

```
 volumes:
 - ~/.cache/httr2:/root/.cache/httr2 # [local path]:[container path]
```

Note that the path returned by `rappdirs::user_cache_dir("httr2")` is the local path and is added before the `:`. If the action is run on a server (which is the use case that makes most sense), one needs to first copy the token there (e.g., copy it to `/srv/amcat/token` and then link this folder to `/root/.cache/httr2` in the container).

3 Conclusion

A crucial criterion for the infrastructure developed by OPTED is scientific transparency to facilitate replication, collaboration, and efficient re-use of academic contributions in terms of data collection and tool creation. The features in the AmCAT suite of packages enable researchers to make data available in ways that conform with copyright, privacy, or other concerns. Following the ideas developed within the concept of non-consumptive research, we can enable new analyses or replication effectivly without sharing access to the full material.