1 Pre Installations

To install Scicopia clone the git repository, change to the project main directory and install the requirements.

```
git clone https://github.com/pikatech/Scicopia
cd Scicopia
pip install -r requirements.txt
```

Scicopia uses two database systems to store the data needed. Firstly ArangoDB[1] to store bibliographic data and metadata of the documents, user data and link data for the graph. Secondly Elasticsearch[2] for the search function. Accordingly it is necessary to install and set up these databases, load data into them (more in Pre Processing) and run them before running Scicopia.

[1] https://www.arangodb.com/
[2] https://www.elastic.co/

2 Configuration

The Configuration is stored in the `config.json` located in the main directory of the project. It defines the used database connections and collections, logging information and some other parameters. The format of the configuration is a dictionary. An empty `dummyconfig.json` to fill in and rename is included.

– Flask –
"secret_key": str, secret key for Flaskapp

– Elasticsearch –
"es_hosts": list, hosts to connect to
"index" : str, name of database
"suggestions" : str, name of database for autocompletion
"fields": list, fields to load into Elasticsearch, can be used for fieldspecific search, recomendet fields: "title", "author", "abstract", "auto_tags"

– ArangoDB –
"arango_url": str, optional, url to connect to
"username": str, ArandoDB username
"password": str, ArangoDB password
"database": str, database with all used collections
"documentcollection": str, collection with documents
"pdfcollection": str, collection with pdfs of documents
"usercollection": str, collection with users
"nodecollections": list, optional, collections with graphnodes
"edgecollections": list, optional, collections with graphedges

– Mail –
"mailusername": str, Email username
"mailpassword": str, Email passwort
"mailsubjectprefix": str, Email subject prefix
"mailsender": str, Email sendername
"mailserver": str, Email server
"mailport": int, Email port
"mailusetls": bool, use TLS

3 Pre Processing

Before running Scicopia it is needed to load some data to search on into the databases. To do so it is necessary to follow a strict order.

3.1 Load the documents into ArangoDB

For this step use the arangodoc.py located in the scicopia directory. It will import the documents in their various formats into the document collection. The database itself and the collection therein will be created, if they do not exist already.
Execute

```
python -m scicopia.arangodoc [parameters]
```

from the root directory. The only mandatory parameter is the type of the input data.
There are parsers for BibTeX, PubMed XML, ArXiv OAI-MPH format and GROBID TEI included in the project. More can be added by supplying a module with a parse() function as shown in the scicopia/parsers directory. LaTeX commands and formatting will be automatically translated into proper Unicode.
The other parameters are optional:
Some take arguments:
`--path`, default="": str, path to the document directory
`-c`, `--compression`, default="none": str, type of compression, supported: gzip, zstd, bzip2
`--batch`, default=1000: int, Batch size of bulk import
`-p`, `--parallel`: int, distribute the computation on multiple cores
`--cluster`: str, distribute the computation onto a cluster

Others act as flags:
`--pdf`: PDFs with same name in same directory as the documents will be imported in the `pdfcollection` (encoded as Base85)
`-r`, `--recursive`: Recurse into subdirectories
`--update`: to update already stored documents (not including PDFs)

3.2 Use Scicopia-tools

There are a few functions to edit the stored documents in the separate Scicopia-tools project https://github.com/pikatech/Scicopia-tools. It

is recommended to use the same `config.json`.

```
python -m scicopia_tools.arangofetch [parameters]
```

You must choose which feature to use and can run the computations in parallel like in `arangodoc.py`.
The implemented features are "auto_tag" and "split"

`auto_tag`: works on the "abstract" to create a list of key phrases that can be used as index terms
`split`: works on the "abstract" to create a list of begin and end indices of the sentences. Without this list, abstracts will *not* be loaded to Elasticsearch. The splits are used to restrict the returned context of Elasticsearch fragments to one sentence each.

3.3 Load Arango data into Elasticsearch

In this step the fields defined in the `config.json` will be copied from ArangoDB to Elasticsearch by using `docimport.py` from the root directory. The search index will be created, if it doesn't exist already.

```
python -m scicopia.elastic.docimport [parameter]
```

There is an optional parameter:
`-t`, `--recent`, default=0: int, only documents that are more recent than this timestamp will be copied

There are a few other features in Scicopia that also need collections in ArangoDB.

3.4 Autocompletion

The autocompletion feature suggests words in real-time based on last two words/tokens of the search form and the data used to create the Suggestion Completion index.
To extract the autocompletion terms use the `ngrams.py` from the Scicopia-tools project. It will use the abstracts of the documents saved in arangoDB for this.
Call it from main directory with

```
python -m scicopia_tools.compile.ngrams [parameter]
```

The mandatory parameter is the name of the output, a Zstandard-compressed archive.
The other parameters are optional:
`-n`, default=2-3: str, the order of the n-grams. use single number x or range x-y
`--threshold`, `-t`, default=0: int, a threshold for n-gram frequencies to be kept
`--patterns`: use a spaCy matcher to extract bigrams. Can only be used for $1 \leq n \leq 5$
`--weighting`: re-weight the frequencies by their n-gram lengths

To import the data, run `suggestions.py` from the Scicopia main directory via

```
python -m scicopia.elastic.suggestions [parameter]
```

The only parameter is the name of the archive containing the n-grams.
It will be imported to the Elasticsearch index defined in `config.py` as "suggestions".

3.5 User administration

The user information is saved in the `usercollection`. If it doesn't exists, an empty one will be created, when the flask server is started.

3.6 Graph features

For the graph features it is necessary to create collections with the nodes and edges of the graph and change the code in `scicopia/app/graph/customize.py` to work with the new attributes, especially color and zpos. Pay attention to the comments. The examples in `scicopia/app/graph/customize_dummy.py` use the "World Graph" example created by ArangoDB. If the graph collections are not defined, the features are disabled. If there is a problem to load the graph data from ArangoDB, e.g. because the defined collections don't exist, an error page will be shown instead.

3.7 Citation graph

The citation graph is a graph created by

```
python -m scicopia.graph.citations
```

It uses the documents from `documentcollection` in arangoDB to create a graph of all the documents containing the `citing` attribute. The graph can be imported with the `documentcollection` as `nodecollection` and "Citations" as `edgecollection`.

4 Running

To run Scicopia it is necessary to run the ArangoDB and Elasticsearch servers first. To run Scicopia on a development server, call the built-in server from the main directory.

```
set FLASK_APP=scicopia/flask_main
flask run
```

For a production setup, it is possible to use a WSGI server with:

```
waitress-serve --host=localhost --call scicopia.flask_main:wsgi
```

The optional parameters are shown with

```
waitress-serve --help
```

and will not explained here, except for setting the port, which is `--port=`. The address to connect to with the browser will be shown in the console after running the server.