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 databasesystems to store the data needed. Firstly ArangoDB[1] to store bibliographic data and meta data of the documents, user data and data for the graph. Secondly Elasticsearch[2] for the search funktion. Accordingly it is necessary to install and set up these databases, load data into them (more in **??**) and run them before running Scicopia.
It is possible to use a different database instead of ArangoDB but it would need many changes in the code to adjust to the new database. It is not recommendet to replace Elasticsearch but it could also be possible.

[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 databaseconnections and collections, loggin informations and some other parameters. The format of the config is a dictionary. An empty dummyconfig.json to fill 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 nessesary to follow an strict order.

3.1 Load the documents into ArangoDB

For this step use the arangodoc.py locatet in the scicopia directory. It will store the data from the documents in the documentcollection. The database and collection will be createt if not existing.
Run from main directory with

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

The only must parameter is for the type of the input data
There are parsers for bibtex, pubmed, arxiv and grobid data included in the project, but it is possible to add more.

The other parameters are optional:
some need other 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

some stay alone:
$--pdf$: pdfs with same name in same directory as the documents will be stored in the pdfcollection
$-r$, $--recursive$: subdirectories will be parsed
$--update$: to update already stored documents (PDFs not included)

3.2 Use Scicopia-tools

There are a few functions to edit the stored documents in the separate Sicopia-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 the used feature and could use the parallel option like in arangodoc.py.
The implemented features are “clean”, “auto_tag” and “split”

“clean”: removes artefacts like Latexcode, works on the “abstract”, “title”, “author” and “fulltext” attributes. It is recomended to use it first because it changes the abstract where the other features are working on.
“auto_tag”: works on the “abstract” to creates a keywordslist
“split”: works on the “abstract” to creates a list with beginn and end index of sentences. Without this list abstracts will NOT be loaded to Elasticsearch.

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 main directory. The database will be created if not existing.

```
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 suggests words to search based on the input of the searchfield and the data used to create the autocompletiondata.
To create the autocompltiondata use the ngrams.py from the Scicopia-tools Project, it will use the abstracts of the documents saved in arangoDB.
Run from main directory with

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

The must parameter is for name of the outputfile.
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 n $\leq$ 5
− − *weighting*: re-weight the frequencies by their n-gram lengths

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

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

The only parameter is for the name of the created file.
It will be imported to the Elasticsearch index defined in config.py as "suggestions".

3.5 Useradministration

The Userdata is saved in the Usercollection. If it doesn't exists, an empty one will be created while starting the flask server.

3.6 Graphfeatures

For the Graphfeatures it is necessary to create collections with the nodes and edges from 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 Graphcollections are not defined the features are disabled. If there is a problem to load the graphdata from ArangoDB e.g. because the defined collections don't exist, an errorpage will be shown instead.

3.7 Citationgraph

The Citationgraph is a graph created by

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

It uses the documents form documentcollection in arangoDB to create a graph using 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 Elastcsearch Servers first. To run Scicopia on a Developeserver use from the main directory

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

For production 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.
The addresse to connect with the browser will be shown in the console after running the server.