RISIS-KNOWMAK Classification Service

User Guide

Adam Funk

22 May 2020

RISIS

RESEARCH INFRASTRUCTURE FOR SCIENCE AND INNOVATION POLICY STUDIES

GATE

The University Of Sheffield.

Contents

1. Overview

The software provides a REST service on GATE's servers with a POST endpoint which accepts documents, classifies them according to the RISIS-KNOWMAK ontology, and returns classification and keyword data in JSON.

For each POST, the request body is turned into a GATE document and run through a GATE application drawn from a multithreaded pool. When the application finishes, the service picks the classification and keyword data out of the document and returns them in the response body, using the JSON format described in §3.

2. Input

2.1. Endpoints

We expect the following POST endpoints to be used:

- `http://services.gate.ac.uk/knowmak/classifier/project/<DOCID>`
- `http://services.gate.ac.uk/knowmak/classifier/project`
- `http://services.gate.ac.uk/knowmak/classifier/patent/<DOCID>`
- `http://services.gate.ac.uk/knowmak/classifier/patent`
- `http://services.gate.ac.uk/knowmak/classifier/publication/<DOCID>`
- `http://services.gate.ac.uk/knowmak/classifier/publication`

but in fact, the service will accept a POST request to any URL in the following format

- `http://services.gate.ac.uk/knowmak/classifier/<DOCTYPE>/<DOCID>`

where `<DOCTYPE>` and `<DOCID>` are valid URL path elements. The `DOCTYPE` is passed as a string to the GATE application, which will use it for conditional processing to handle *project*, *patent*, and *publication* documents slightly differently (by increasing the scores of keyword matches that are especially relevant to the document type). The system is designed to accept any `DOCTYPE` string in order to allow for future expansion; an unexpected type is flagged with a warning in the output (see §3).

2.2. Data formats

The service can accept any format that GATE can process by default[1], in particular:

- plain text,
- HTML (including XHTML),
- XML and SGML,
- RTF, and
- some Microsoft Office and OpenOffice formats.

The service will work much more reliably if the client supplies a correct `Content-Type` header (including the charset).

[1] `https://gate.ac.uk/sale/tao/splitch5.html#sec:corpora:formats`

3. JSON annotation output format

3.1. Example


```
{
  "classification": {
    "http://www.gate.ac.uk/ns/ontologies/knowmak/applied_immunology": {
      "boostedBy": "",
      "keywords": {
        "virus": {
          "kinds": ["generated"],
          "offsets": [753, 760, 819, 826],
          "score": 0.37968825194608924
        }
      },
      "score": [0.37968825194608924, 0.44063387427256007],
      "topicID": "18",
      "unboostedScore": [0.37968825194608924, 0.44063387427256007]
    },
    "http://www.gate.ac.uk/ns/ontologies/knowmak/gene_delivery": {
      "boostedBy": "",
      "keywords": {
...
        }
      },
      "score": [0.3934696359661883, 0.4566273757898954],
      "topicID": "58",
      "unboostedScore": [0.3934696359661883, 0.4566273757898954]
    },
...
  },
  "doc_length": 1857,
  "doc_size": 347,
  "doc_type": "publication",
  "doc_type_applied": "publication",
  "error": "_none_",
  "identifier": "antibiotics",
  "internalID": "d6d44d08-d32e-415e-b9e0-d1d0963e35ea"
}
```


3.2. Explanation

The output is a JSON map with the following keys.

- `classification` The value is a map whose keys are ontology classes; each value is a map with the following keys.
  - `score` A list of two float values, showing the scores with boosting from superclasses for this class (without and with PMI boosting).

  - `unboostedScore` A list of two float values, showing the scores without boosting from superclasses for this class (without and with PMI boosting).
  - `boostedBy` This has a string value with the URI of the boosting superclass. If the value is the empty string, no direct superclass was found and the superclass-boosted scores are the same as the unboosted ones.
  - `keywords` The value is a map with a keyword string (which can be a multi-word phrase) as each key and a map with the following keys for each value.
    * `kinds` The value is a list of strings with the *flag* values for the keyword.
    * `score` The value is a float showing that keyword's contribution to the class score (base score multiplied by number of occurrences, without PMI or superclass boosting). Note that the same keyword may contribute to more than one class and can have different scores for different classes.
    * `offsets` The value is a list of integers representing the positions of the keyword matches in the document. The list always contains an even number of integers, alternating between start and end offsets.

      For example, `[300, 307, 350, 357]` indicates two matches from 300 to 307 and from 350 to 357. The length of the match may not be exactly the same as the length of the lemmatized keyword (e.g., *antibiotics* in the document matching *antibiotic* as a keyword).

- `doc_size` The number of GATE *Token* annotations in the document as an integer.
- `doc_length` The number of characters of plain text in the document content as an integer.
- `doc_type` The string value shows the `DOCTYPE` parameter passed in the endpoint URL (see §2.1).
- `doc_type_applied` The string value shows the `DOCTYPE` parameter if it was validated by the service[2] or `_none_` if it was invalid.
- `error` The string value shows an error encountered within the GATE application while processing the document, or `_none_` for no error.
- `identifier` The string value shows the `DOCID` parameter if it was provided in the endpoint URL, or `_none_`.
- `internalID` The string value shows an internal identifier generated within the service and used for debugging (this ID appears in the server log files).

[2] As mentioned above, current valid values are *paper*, *project*, and *publication*.

4. Code examples

4.1. Curl

Processing an individual file:

```
curl  -X POST  --data @my_file.txt  -o output.json \
      --header "Content-Type: text/plain"             \
      http://services.gate.ac.uk/knowmak/classifier/project
```

Processing standard output from another command:

```
my_command  |  curl  -X POST  --data-raw -  -o output.json \
                  --header "Content-Type: text/plain"   \
                  http://services.gate.ac.uk/knowmak/classifier/project
```

4.2. Python

We recommend the Python Requests library[3], as shown in this example. Note that the method returns a Python `dict` (converted automatically from JSON by the library).

```
import requests

base_url = "http://services.gate.ac.uk/knowmak/classifier"

def use_gate_service(text, doc_type, content_type='text/plain',
                     doc_id=None):
    if doc_id:
        url = '/'.join([base_url, doc_type, doc_id])
    else:
        url = '/'.join([base_url, doc_type])
    response = requests.post(url, data=text,
                             headers={'Content-Type': content_type})

    status = response.status_code
    if status != 200:
        raise ValueError("HTTP error " + status)

    return response.json()
```

4.3. Java

Various HTTP client libraries are available, including the Spring RestTemplate[4].

[3] http://docs.python-requests.org/

[4] https://docs.spring.io/spring-boot/docs/current/reference/html/boot-features-restclient.html