PaNOSC Federated Search Results Scoring API

Compiled and reviewd by ESS team
(Max Novelli, Henrik Joansson , Fredrik Bolmsten, Tobias Richter)
v 2022-01-13

Table of Contents

Introduction

This documents list all the endpoints available through the Scoring API. The endpoints are organized in sections grouping endpoints pertaining to the same type object or actions.
Please refer to the document PaNOSC Federated Search Results Scoring document for more detailed description of the theory behind the Scoring service and for the definitions of key concepts.

Items

Items are the scoring PaNOSC datasets and documents that are going to be used in the weight computation and, later on, for the scoring. We decided to call them ***items***, instead of using the most common term ***documents***, because in the PaNOSC project the word document is a specific type of entity which is a subset of all the possible entities that we can possibly perform the scoring on.

Model

The item model has the following fields:

- *id*
  id/pid of the item provided by catalogue system
- *group*
  string defining the group this item belongs to. Example: datasets or documents. Default: *default.*
- *fields*
  scoring information. It can contain a string or a complex nested json object. It is named fields as in most cases is a set of relevant fields extracted from catalogue system. The minimum set of scoring information suggested are the fields returned by the PaNOSC federated search.

Endpoints

GET /items

Returns the complete list of items saved in the database
Parameters:

- limit: number of items returned. Default: (int) 1000
- offset: number of items to remove before extracting the number of elements requested with *limit*. Default: (int) 0

Returns:

- Array of items. Each items has the same structure as reported in the model section.

GET /items/count

Returns the total number of items across all groups.
Parameters:

- *none*

Returns:

- Json objects with the following structure:
  `{ count: (int) }`

GET /items/{id}

Returns the item with id matching {id}. It returns None and 404 if the item is not found. {id} is the same id provided for the item by the catalogue system.

Parameters:

- {id}: (string) item id as it is in the catalogue system.

Returns:

- Requested item with structure as in model.

POST /items

Insert one or more new items with their scoring information that we would like to be score in the database.
Parameters:

- Single item or array of items. Each items should match the structure defined in the model section.

Returned:

- Status code: 201
- Operation summary with the following structure:
```
{
 ‘success’ : True,
 ‘items_created’ : (int),
 ‘items_ids’ : (list of strings)
}
```

DELETE /items/{id}

Delete the item with id {id}.
Parameters:

- {id}: (string) item id as it is in the catalogue system.

Returns:

- Status code: 200
- Operation summary with the following structure:
```
{
 ‘successful’ : True,
 ‘items_deleted’ : (int)
}
```

PUT /items/{id}

Update the whole item with id {id}
Parameters:

- {id}: (string) item id as it is in the catalogue system.
- New values of the fields of the item with the following structure
```
{
 ‘group’ : (string),
 ‘fields’ : (object)
}
```

Returns:

- Status code: 200
- Operation summary with the following structure:
```
{
 ‘successful’ : True,
 ‘items_updated’ : (int)
}
```

PATCH /items/{id}

Partial update of the item with id {id}
Parameters:

- {id}: (string) item id as it is in the catalogue system.
- New values of the fields of the item that needs to be updated. Structure required:
  ```
  {
   ‘group’ : (string)[optional],
   ‘fields’ : (object)[optional]
  }
  ```

Returns:

- Status code: 200
- Operation summary with the following structure:
  ```
  {
   ‘successful’ : True,
   ‘items_updated’ : (int)
  }
  ```

Weights computation

The weights computation status is saved in the database and updated according to the user requests and also by the background processes.

Model

The weights computation model contains the following fields:

- *id*
  UUID assigned by the system when a new weights compute request has been received
- *requested*
  timestamp when the weight computation request was last received. It is empty only if the system has not received any weight computation request since it was initialized.
- *started*
  timestamp when the background process has started the weights computation. It is empty if a new request has come in and the background process has not been spawn yet.
- ended
  timestamp when the background process has completed the weights computation. It is empty if a new request has been received and not served yet, or the computation is on-going.
- progressPercent
  Progress in percent of the weights computation procedure. 0% means that the request has been received but not served yet. It is 100% when the computation has been completed and the old weights have been deleted from the database.
- progressDescription
  current action performed during the computation procedure. Used mostly during debugging and system, setup.
- inProgress
  boolean field indicating if a weight computation process is active. This fields indicates that the weights are being updated and might not be consistent.

Endpoints

GET /compute

Returns information about the weight computation status
Parameters:

- *none*

Returns:

- Status code: 409
- Weight computation status with the following schema:

```
{
 requested           : (str)[optional] timestamp ISO 8601 yyyymmddTHHMMSS.FFF+ZZ
 started             : (str)[optional] timestamp ISO 8601 yyyymmddTHHMMSS.FFF+ZZ
 ended               : (str)[optional] timestamp ISO 8601 yyyymmddTHHMMSS.FFF+ZZ
 progressPercent     : (int)
 progressDescription : (str)
 inProgress          : (boolean)
}
```

or

- Status code: 500
- Error message with the following schema:

```
{
  message: (str)
}
```

POST /compute

Triggers a new weight computation.
Parameters:

- none

Returns:

- Status code: 404
- Weight computation status with the following schema:

```
{
 requested           : (str) timestamp ISO 8601 yyyymmddTHHMMSS.FFF+ZZ
 started             : (str) ””
 ended               : (str) ””
 progressPercent     : (int) 0
 progressDescription : (str) “”
 inProgress          : (boolean) True
}
```

or

- Status code: 500
- Error message with the following schema:

```
{
  message: (str)
}
```

Weights management

Weights endpoint only provides a way to retrieve the list of all weights or by group or the count. Weights cannot be modified from these endpoints.

Model

Weights are the output of the weight computation process. They are save individually in the database according to the following model:

- *id: (str)*
  UUID automatically assigned by the weights computation process
- *term: (str)*
  Term extracted from the scoring information of one or more items.
- *ItemId: (str)*
  UUID of the item where this terms appears with the associated weight
- *itemGroup: (str)[optional]*
  Group of the item where this term has been extracted from. It is duplicated to facilitate querying. Default: *default*
- *timestamp: (datetime)*
  Timestamp when this weight has been computed.
- *value: (float)*
  Value of the weight associated with the pair (item,term)

Endpoints

GET /weights

Return all the weights stored in the service related to one of the groups.
Parameters:

- group: (str)[optional]

Returns:

- Status code: 200
- list of weights. Each weight will follow the model above.

GET /weights/count

Returns the total number of non-zero weights present in the database.
Parameters:

- group: (str)[optional]

Returns:

- Status code: 200
- number of weights with the following structure:

```
{
 count: (int)
}
```

Terms management

Model

No database model is associated with terms. Terms are derived from the weights models.

Endpoints

GET /terms

Returns the list of terms extracted from all the items.
Parameters

- Group: (str)[optional] Specify the group we would like to retrieve terms from.

Output:

- List of terms with the following structure:

```
{
 term: (str)
 numberOfIterms: (int) number of items where the terms is found
 numberOfGroups: (int) number of groups where this terms is found
}
```

Score computation

Model

No database model is associated with scores. They are computed using weights but they are persistent in the database.

Endpoints

POST /score

Returns the score for the items selected according to the query submitted.
Parameters:

- *query (str)*
  sentence or list of key words to use in the relevancy scoring
- *itemsId (list of str)[optional]*
  list of items id that needs to be scored in this request. Usually this list is provided by the catalogue system based on the hard filters provided. If no item ids are provided, all the items present in the database will be scored. Default: none.
- *group: (str)[optional]*
  group of items to be scored. If not provided, all type of items from all groups will be scored. If provided, only the items of the selected group will be scored. If items list is provided, it will be filtered according to the specified group. Default: none
- *limit: (int)[optional]*
  number of item to return after the scoring has been done and items have been sorted according to scores. All the scored items will be returned if not provided or it has a negative value. Default: *-1*

Returns:

- *request (object)*
  object containing the request as it was posted. Please check the previous section for details
- *query:*
  - *query (str)*
    query string as it was provided in the request
  - *terms (list of str)*
    list of the term extracted from the query submitted. Please consult related documentation on the details on how term are extracted from query and text in general.
- *scores (list):*
  list of tuples, one for each item scored, with the following structure:
  - *itemId (str)*
    uuid of the item scored
  - *score (float)*
    relevancy score of the item in the query space
  - *group (str)*
    group of the item which this score pertains to
- *dimension (int)*
  number of items scored returned

- *computeInProgress (boolean)*
  indicates if weigths computation is in progress and weights might not be consistent
- *started (timestamp)*
  timestamp of when the score computation has started
- *ended (timestamp)*
  timestamp of when the score computation has ended