User guide for the ARROW API

1. Introduction

This user guide explains how to use the ARROW API with the basic configuration. It declares all functions of the API, how to use them and give examples for requests and their results.

2. Requests

The ARROW API is based on REST principles: data resources are accessed via standard HTTP requests in UTF-8 format to an API endpoint. The API is running on a server hosted by amazon AWS. To use this API you have to send a POST request to the following address:

http://ec2-52-212-74-103.eu-west-1.compute.amazonaws.com:4000/api/score

3. Input scheme

All data is received as a JSON object that are based on the following scheme:

```
{
    "file": {
      "title": "(required) string ",
      "type": "(required) string png/jpeg",
      "created_at": "(required) unix timestamp",
      "user": "(required) id string",
      "description": "string that was sent with the upload"
    },
    "tasks": [
      {
        "title": "(required) string",
        "created_at": "(required) unix timestamp",
        "due_date": "unix timestamp",
        "created_by": "(required) id string",
        "last_updated_by": "id string",
        "last_updated_at": "unix timestamp",
        "assignees": ["userid1", "userid2", "..."],
        "description": "string",
        "location": "..."
    },
    "..."]
}
```

The fields marked with '(required)' are necessary for the comparison. If one or more of these fields are missing the API will return an exception.

4. Functions

There are three important concepts that facilitate assigning a score to tasks representing the degree to which they match an uploaded file and its metadata: The Score Manager, one or more Plugins and an Aggregator.

A Plugin is a function that takes two arguments - a file object that contains meta data, for example the filename, size, time of upload and/or the file contents, and a task object that contains meta data related to the task, for example the task name. It returns a floating point numeric score in the range 0.0 to 1.0 which describes the degree in which the file and the task are correlated in the aspect that this plugin is focused on. There are 2 Plugins available:

1. **The similar-context Plugin:**
   This plugin different types of texts like descriptions or titles of files and tasks. If the content of the two texts are similar but have different descriptions, the result would be about 1.0

2. **The close-time Plugin:**
   It checks the time, when both objects were uploaded (or updated) and if the upload times are far away from each other the plugin would return 0.0. Otherwise if the objects are uploaded at the same time the result would be 1.0.

An **Aggregator** is a policy that combines a set of scores that were previously assigned to a task by multiple Plugins into a single final score value. For example, if the score of the close-time Plugin is 1.0 and the score of the similar-title Plugin is 0.0 the combined value would be 0.5.

The purpose of the **Score Manager** is to provide the entry point for a scoring request, delegate the data to multiple Plugins, and combine their individual scores using an Aggregator.

5. Example

Request:

```
{
    "file": {
      "title": "Cafe abc",
      "type": "jpeg",
      "created_at": "1479755100",
      "user": " 5hj34thtr",
      "description": " Great location for a meeting"
    },
    "tasks": [
      {
        "title": " find a location",
        "created_at": "1479754800",
        "due_date": "1479766305",
        "created_by": "ikgDG94s",
        "description": "Find a location for the next meeting"
    },
     {
        "title": " Check your mails",
        "created_at": "1379754800",
        "due_date": "1454353454",
        "created_by": "dfgj2s334",
        "last_updated_by": "ikgDG94s",
        "description": "Check your mails before you leave."
    }

  ]
}
```

6. Configuration

The Basic-Configuration of this API makes 5 different comparisons:

1. **Similar Title:**
   Compares the title of the file with all task titles

2. **Context File Timestamp – Tasks Timestamp:**
   Compares the timestamps of file and tasks with a time limit of 600 seconds

3. **Context File Timestamp – Tasks Timestamp (Long):**
   Compares the timestamps of file and tasks with a time limit of 3000 seconds

4. **Context File Description – Task Title:**
   Compares the description of a file with the title of tasks

5. **Context File Description – Task Description:**
   Compares the description of a file with the description of tasks

There is also the possibility to configure the API by themselves, just by sending an own configuration with the Post Request. To create an own configuration for this API, define a name of what is compared, the plugin that is used and the input fields which are compared. Optionally there is the possibility to add several parameters. An example could look like this:

```
"plugins":{
    "context-file-description-task-description": {
      "use": "similar-context-plugin",
      "inputs": [" file.description", "tasks[].description"],
      "params": { "extractKeywords": true }
    }
```

This configuration is used to compare the description of file and tasks ("inputs": [" file.description", "tasks[].description"]) by keywords ("params": { "extractKeywords": true }) with the similar context ("use": "similar-context-plugin") plugin.

After defining the used plugins, the score aggregator must be configured. There are 3 different types of aggregators:

1. **The Mean Aggregator:**
   This aggregator combines scores by calculating the average of all scores

2. **The Largest Aggregator:**
   Is an aggregator that combines scores by choosing the biggest score out of all scores.

3. **The Weighted Mean Aggregator:**
   Combines scores by calculating the weighted average of all scores

Example:

```
"config":{
        "aggregator": "Mean",
        "plugins":{
                "context-file-description-task-description": {
                "use": "similar-context-plugin",
                "inputs": [" file.description", "tasks[].description"],
                "params": { "extractKeywords": true }
        }
}
```