Advanced recommendation parameters

The recommendation engine has a lot of internal parameters for fine tuning the underlying algorithm. For example, when determining the relevance of any given article, its popularity and freshness can alter its score (popular and recent articles score higher). The weight of popularity and freshness when computing the article score can be controlled by setting the respective parameters (`beh.weight.popularity` and `beh.weight.freshness` respectively) to better suit particular needs.

There are many more parameters to adjust, see the reference at the end of this document.

Parameters can be set per individual request or globally, for all requests.

Passing parameters for recommendation request

Both `/recommend/contextual` and `/recommend/behavioural` calls can be called with `GET` and `POST` HTTP methods. When `POST`-ing, some advanced parameters could be passed for fine tuning the results. `Content-type` for `POST` requests should be **`application/json`** and the request body should contain a simple JSON object with parameter name and value pairs. For example:

```
POST /recommend/contextual?contentid=<existing_id>
Content-type: application/json

{
  "beh.weight.popularity": 0.2,
  "beh.weight.freshness" : 0.8
}
```

Setting parameters globally

Getting and setting global parameter values goes through the `/weight` endpoint. `GET` returns a list of parameters and their type, description and current value

```
GET /weights

[
  {
    "name": "mlt.minwl",
    "type": "INTEGER",
    "value": 1,
    "description": "minimum word length below which words will be ignored"
  },
  // ... more parameters here
]
```

`PUT` or `POST` set values. As in with parameters per recommendation call, the `Content-type` of the request should be **application/json** and the request body should contain a simple object with name -> value pairs:

```
POST /weights
Content-type: application/json

{
  "beh.weight.popularity": 0.2,
  "beh.weight.freshness" : 0.8
}
```

Parameter reference

All recognized parameters, values below are the defaults:

```
[
  {
    "name": "beh.articleDampingFactor",
    "type": "INTEGER",
    "value": 10,
    "description": "when a user haven't read many articles, the weight of his profile
is lowered. This parameter sets the minimum number of articles a user must read before
his profile gains full weight"
  },
  {
    "name": "beh.boost.expr",
    "type": "STRING",
    "value": null,
    "description": "an optional custom expression for boosting the score of articles
for behavioural recommendation. If omitted, documents are boosted based on recency,
popularity and covisitation - and the relative weight of all three can be tweaked with
the appropriate beh.weight.XXX parameter. Solr function query syntax:
http://wiki.apache.org/solr/FunctionQuery"
  },
  {
    "name": "beh.weight.freshness",
    "type": "DOUBLE",
    "value": 1.0,
    "description": "relative weight of the freshness/recency of an article (only when
beh.boost.expr is missing)"
  },
  {
    "name": "beh.weight.popularity",
    "type": "DOUBLE",
    "value": 1.0,
    "description": "relative weight of an article's popularity (only when
beh.boost.expr is missing)"
  },
  {
    "name": "beh.weight.covisitation",
    "type": "DOUBLE",
    "value": 1.0,
    "description": "relative weight of article covisitation (only when beh.boost.expr
is missing)"
  },
  {
    "name": "beh.weight.moreLikeThis",
    "type": "DOUBLE",
    "value": 1.0,
    "description": "weight of the article content similarity portion when calculating
behavioural recommendation"
  },
  {
    "name": "beh.weight.userProfile",
    "type": "DOUBLE",
    "value": 1.0,
    "description": "weight of the user profile similarity portion when calculating
```

```
behavioural recommendation"
  },
  {
    "name": "beh.alpha",
    "type": "DOUBLE",
    "value": 0.95,
    "description": "parameter that controls the stickiness/momentum of update of a
user profile's transition matrix"
  },
  {
    "name": "beh.gamma",
    "type": "DOUBLE",
    "value": 0.98,
    "description": "parameter that controls the stickiness/momentum of update of a
user profile's static term matrix"
  },
  {
    "name": "mlt.fl",
    "type": "STRING",
    "value": "title,summary,content,tags,keyphrases",
    "description": "fields to use for content similarity"
  },
  {
    "name": "mlt.mintf",
    "type": "INTEGER",
    "value": 1,
    "description": "minimum Term Frequency - the frequency below which terms will be
ignored in the source doc"
  },
  {
    "name": "mlt.mindf",
    "type": "INTEGER",
    "value": 1,
    "description": "minimum Document Frequency - the frequency at which words will be
ignored which do not occur in at least this many docs. "
  },
  {
    "name": "mlt.minwl",
    "type": "INTEGER",
    "value": 1,
    "description": "minimum word length below which words will be ignored"
  },
  {
    "name": "mlt.maxwl",
    "type": "INTEGER",
    "value": 2000000000,
    "description": "maximum word length above which words will be ignored"
  },
  {
    "name": "mlt.boost",
    "type": "BOOLEAN",
    "value": false,
    "description": "set if the query will be boosted by the interesting term
relevance. "
  },
  {
    "name": "mlt.maxqt",
    "type": "INTEGER",
    "value": 100,
    "description": "maximum number of query terms that will be included in any
generated query. "
  },
  {
    "name": "mlt.mintf.perfield",
    "type": "STRING",
    "value": "content^1 tags^1 title^1 keyphrases^1 summary^1",
    "description": "mlt.mintf per field using format \"field1^mintf1 field2^mintf2
...\""
  },
  {
    "name": "mlt.boostexpr",
    "type": "STRING",
    "value": "recip(ms(NOW,published),3.858024691358025e-10,1,1)",
    "description": "boosting expression for content recommendation"
  },
  {
    "name": "mlt.interestingTerms",
    "type": "STRING",
    "value": "details",
    "description": "one of: \"list\", \"details\", \"none\" -- this will show what
\"interesting\" terms are used for the MoreLikeThis query. These are the top tf/idf
terms. NOTE: if you select 'details', this shows you the term and boost used for each
term. Unless mlt.boost=true all terms will have boost=1.0 "
  },
  {
    "name": "mlt.qf",
    "type": "STRING",
    "value": "content^1.0 tags^2 title^0.25 keyphrases^2 summary^2",
    "description": "query fields and their boosts using format \"field1^boost1
field2^boost2 ...\". These fields must also be specified in mlt.fl"
```

```
  }
]
```