SSECore Documentation

Release latest

February 24, 2016

Contents

1 Introduction **1**
1.1 Background and Detail 1

2 User Guide **3**

3 Programmer Guide **5**

4 Installation and Administration Guide **7**

CHAPTER 1

Introduction

Welcome the User and Programmer Guide for the Social Semantic Enricher Generic Enabler. The online documents are being continuously updated and improved, and so will be the most appropriate place to get the most up to date information on using this interface.

1.1 Background and Detail

This User and Programmers Guide relates to the Social Semantic Enricher GE which is part of the Data Chapter. Please find more information about this Generic Enabler in the following Open Specification.

CHAPTER 2

User Guide

The Social Semantic Enricher GE is a backend component, A user interface was provided, but since the development of the enabler were stopped before the end of the project it was not possible to complete it. Therefore there is no need to provide a user guide. The enabler provides anyway a set of two main APIs. The Core One, which is detailed here is the “Classify” Api, which for a given text, returns a set of concepts, explaining what the text is about, plus some related info for each concept.

CHAPTER 3

Programmer Guide

The Classify API as Input and via text processing and Lucene Analysis returns a URI list. When a Text comes as input, some lemmatisation and stemming operations are performed and after that, the SSE compares the input text with ALL of the contexts fields saved into the Lucene Dataset (which means, it compares the text with all the paragraphs in wikipedia containing at least one link). For each context, a similarity metric, offered by the Lucene Technology is computed. The Set of N(which can be specified to the system) URIs corresponding to the contexts with the higher level of similarity is returned. Next We show the input parameters to send, the expected output parameters and a sample request.

- **Input Parameters**
  - text - The Text To classify
  - lang - The Text Language
  - numTopics - The number of topics to extract
- **Output Parameters**
  - uri - dbpedia URI of the resource found
  - label - dbpedia label of the resource found
  - title - dbpedia title of the resource found
  - score - relevance score of the concept in relation to with the text
  - mergedTypes - a string summarizing dbpedia "type" voice of the resource
  - image - Dbpedia internal image of the resource if any
  - wikilink - Link to wikipedia voice related to resource
- Request (application/json)

  { "text": "The final work of legendary director Stanley Kubrick, who died within a week of completing the edit, is based upon a novel by Arthur Schnitzler. Tom Cruise and Nicole Kidman play William and Alice Harford, a physician and a gallery manager who are wealthy, successful, and travel in a sophisticated social circle.", "lang": "en", "numTopics": "7"

  }
- Response 200 (application/json)

  "uri": "http://dbpedia.org/resource/Stanley_Kubrick", "label": "Stanley Kubrick", "title": "Stanley Kubrick", "score": "0.68900007", "mergedTypes": "DBpedia:Person#DBpedia:Http://wikidata.dbpedia.org/resource/Q5#DBpedia:Http://xmlns.com/foaf/0.1/Person#Schema:Person# "image": "http://commons.wikimedia.org/wiki/Special:FilePath/Kubrick_-_Barry_Lyndon_candid.JPG", "wikilink": "http://en.wikipedia.org/wiki/Stanley_Kubrick"

}, { “uri”: “http://dbpedia.org/resource/Eyes_Wide_Shut”, “label”: “Eyes Wide Shut”, “title”: “Eyes Wide Shut”, “score”: “0.6772491”, “mergedTypes”: “DBpedia:Film#Schema:Movie#DBpedia:Wikidata:Q11424#DBpedia:Work#Schema:CreativeWork#DBpedia:Http://www.ontolog “image”: “”, “wikilink”: “http://en.wikipedia.org/wiki/Eyes_Wide_Shut” }, { “uri”: “http://dbpedia.org/resource/Nicole_Kidman”, “label”: “Nicole Kidman”, “title”: “Nicole Kidman”, “score”: “0.6715633”, “mergedTypes”: “DBpedia:Person#DBpedia:Http://wikidata.dbpedia.org/resource/Q5#DBpedia:Http://xmlns.com/foaf/0.1/Person#Schema:Person# “image”: “http://commons.wikimedia.org/wiki/Special:FilePath/Nicole_Kidman_2,_2013.jpg”, “wikilink”: “http://en.wikipedia.org/wiki/Nicole_Kidman” }, { “uri”: “http://dbpedia.org/resource/Arthur_Schnitzler”, “label”: “Arthur Schnitzler”, “title”: “Arthur Schnitzler”, “score”: “0.631234”, “mergedTypes”: “DBpedia:Writer#DBpedia:Artist#DBpedia:Person#DBpedia:Http://wikidata.dbpedia.org/resource/Q5#DBpedia:Http://xmlns.com “image”: “http://commons.wikimedia.org/wiki/Special:FilePath/Arthur_Schnitzler_1912.jpg”, “wikilink”: “http://en.wikipedia.org/wiki/Arthur_Schnitzler” }, { “uri”: “http://dbpedia.org/resource/Dream_Story”, “label”: “Dream Story”, “title”: “Dream Story”, “score”: “0.59502685”, “mergedTypes”: “DBpedia:Book#Schema:Book#DBpedia:Http://purl.org/ontology/bibo/Book#DBpedia:WrittenWork#DBpedia:Work#Schema:Cre “image”: “”, “wikilink”: “http://en.wikipedia.org/wiki/Dream_Story” }, { “uri”: “http://dbpedia.org/resource/Tom_Cruise”, “label”: “Tom Cruise”, “title”: “Tom Cruise”, “score”: “0.5551989”, “mergedTypes”: “DBpedia:Person#DBpedia:Http://wikidata.dbpedia.org/resource/Q5#DBpedia:Http://xmlns.com/foaf/0.1/Person#Schema:Person# “image”: “http://commons.wikimedia.org/wiki/Special:FilePath/Tom_Cruise_by_Gage_Skidmore.jpg”, “wikilink”: “http://en.wikipedia.org/wiki/Tom_Cruise” }, { “uri”: “http://dbpedia.org/resource/Stanley_Kubrick:_A_Life_in_Pictures”, “label”: “Stanley Kubrick: A Life in Pictures”, “title”: “Stanley Kubrick: A Life in Pictures”, “score”: “0.42995054”, “mergedTypes”: “DBpedia:Film#Schema:Movie#DBpedia:Wikidata:Q11424#DBpedia:Work#Schema:CreativeWork#DBpedia:Http://www.ontolog “image”: “”, “wikilink”: “http://en.wikipedia.org/wiki/Stanley_Kubrick:_A_Life_in_Pictures” }]

CHAPTER 4

Installation and Administration Guide

SSE is a tool for classifying and enriching textual documents via Linked Open Data. It uses [Lucene](http://lucene.apache.org/core/) indexes for its classification and enrichment system. To build such indexes use SSE Index Builder project. The core of SSE is the lowest level component that directly interacts with Lucene.

First you need to initialize the settings and the index using the following code:

```
SSEConfig sseConfigFromCache = ConfigCache.getOrCreate(SSEConfig.class); IndexesUtil.init();
```

Once you have initialized SSE's core, as described above, you can invoke *classify()* to classify text, also from code (and not using API described).

```
// // Here text is a String, numTopics is a integer and language // is again a String (typically either "en"
or "it"). // Classifier classifier = new Classifier(language); List<String[]> res = classifier.classify(text,
numTopics);
```

The *classify()* function follows the traditional SSE policy by which large texts are divided in chunks classified separately, and the result is generated merging the classification of each chunk of text.

You can bypass this policy by using the *classifyShortText()* function that directly passes the text to Lucene. Note, however, that depending on the Lucene configuration and on the text length, this call may raise an exception if the resulting Lucene query is too large.

```
Classifier classifier = new Classifier(language); List<String[]> res = classifier.classifyShortText(text,
numTopics);
```