LODtagger

Guide for Users and Developers

Bahar Sateli
René Witte

Release 1.1
October 7, 2016

Semantic Software Lab
Concordia University
Montréal, Canada
http://www.semanticsoftware.info

Contents

About this document

This document contains the documentation for the GATE *LODtagger* component. You can obtain the latest version from http://www.semanticsoftware.info/lodtagger.

License

Support

For question or general comments, please use the online forum at http://www.semanticsoftware.info/forums/tools-resources-forum/durm-corpus-wiki-tools

[1] LGPL3, https://www.gnu.org/licenses/lgpl-3.0.en.html

Chapter 1

LODtagger

The LODtagger is a GATE component that provides entity linking from a document to their corresponding resource on the Linked Open Data (LOD) cloud. LODtagger relies on external tools to perform the actual content tagging and hides the complexity of communicating with LOD taggers, such as DBpedia Spotlight, from the perspective of pipeline developers.

The current version of the LODtagger supports DBpedia Spotlight (see Section 1.1). We plan to integrate additional existing LOD tagging tools in the future.

1.1 DBpediaTagger

The DBpediaTagger component sends the content of a GATE document to a pre-defined DBpedia Spotlight[1] RESTful endpoint and translates the web service results from JSON objects to GATE annotations.

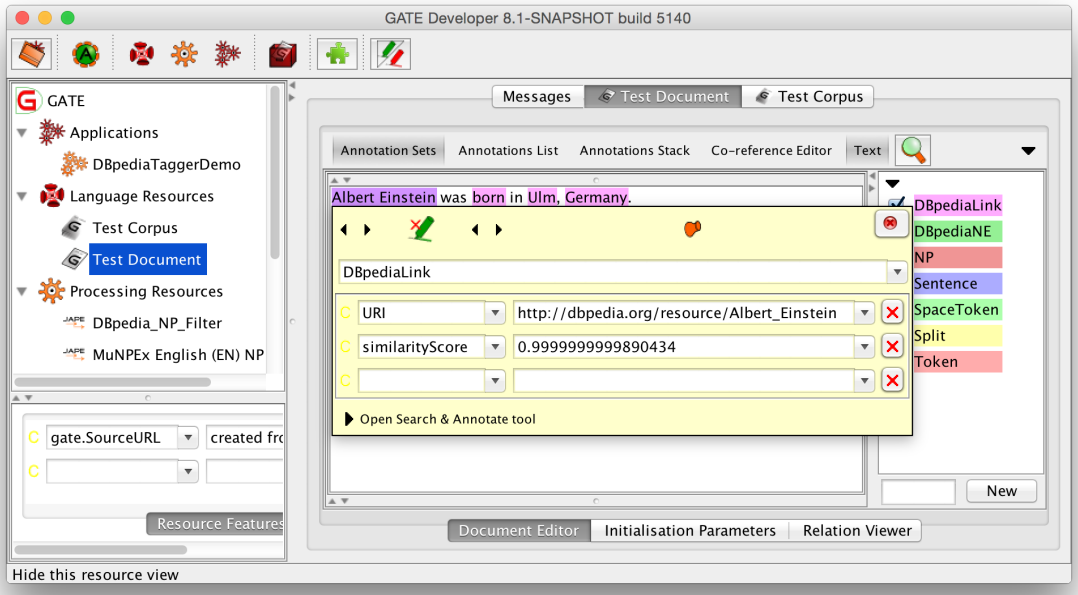

Figure 1.1: Example annotations generated by LODtagger using DBpedia Spotlight

DBpedia Spotlight is an open-source Named Entity Recognition (NER) tool that matches the surface form of words in a text to their entry in the DBpedia knowledge base. Each anno-

[1] DBpedia Spotlight, http://spotlight.dbpedia.org

tated entity is linked to the LOD cloud through a URI, where additional, machine-readable information exists to describe the entity under investigation. Spotlight provides multiple web services, such as *Spotting* (i.e., recognition of entities in text) and *Disambiguate* (i.e., linking of entities to DBpedia knowledge base URIs). In the current release of DBpediaTagger, we support the *Annotate* service, which combines the spotting and disambiguation services. For more information please check the Spotlight web services page.[2]

The DBpediaTagger component allows the user to define the *type* of the annotations generated from the results through a run-time parameter (see Section 1.1.3). Along with the URI, other metadata, such as the confidence of the tool in linking the word to its correct sense is returned to the client. Therefore, for each result received from Spotlight, an annotation is added to the document, which includes several features (Figure 1.1):

URI the URI of the entity in DBpedia knowledge base;

similarityScore a numerical value between 0 to 1, showing the semantic similarity of the entity's in the document form to the corresponding entity described in the knowledge base.

1.1.1 Installation

Prerequisites. To run DBpediaTagger, you must have GATE installed. Please refer to the GATE User Guide[3] for instructions on how to download and install GATE. The automated install method described below requires GATE version 8.1 or better. You also need access to a DBpedia Spotlight endpoint (see Section 1.1.2 for details). The examples and results shown in this document use a local installation of DBpedia Spotlight version 0.7 with a statistical model for English (en_2+2).

PR and Demo Pipeline. The distribution package `creole.zip` comes with the PR and a demo pipeline that you can install directly from within GATE:

1. Start GATE, open the Plugin Manager *(File → Manage CREOLE Plugins...)*.
2. On the fourth tab *(Configuration)*, select a writable directory for the local plugin installations and enable the "Semantic Software Lab" repository.
3. On the third tab *(Available to install)*, you will now find DBpediaTagger. Select it and click on "Apply All".
4. On the first tab *(Installed Plugins)*, you will now find the DBpediaTagger plugin. Select it (either "Load Now" or "Load Always") and click on "Apply All".

To load the example pipeline (Figure 1.2), go to *Applications → Ready Made Applications → Semantic Software Lab → DBpediaTagger Demo Application*.[4]

1.1.2 Usage

The DBpediaTagger component does not rely on any other processing resources in GATE, nor does it require pre-processing of the text. Simply add the DBpediaTagger processing resource to a corpus pipeline and it will annotate the document.

However, in the demo pipeline, we added several other processing resources that help to find noun phrases in text and filter the Spotlight results to named entities, such as "*service-oriented architecture*". The noun phrases in text are detected by the MuNPEx[5] component. A

[2] Spotlight Web Services, https://github.com/dbpedia-spotlight/dbpedia-spotlight/wiki/Web-service
[3] GATE User Guide, https://gate.ac.uk/sale/tao/split.html
[4] Please refer to the GATE user's guide, http://gate.ac.uk/sale/tao for further details on working with pipelines in GATE.
[5] Multi-lingual Noun Phrase Extractor (MuNPEx), http://www.semanticsoftware.info/munpex

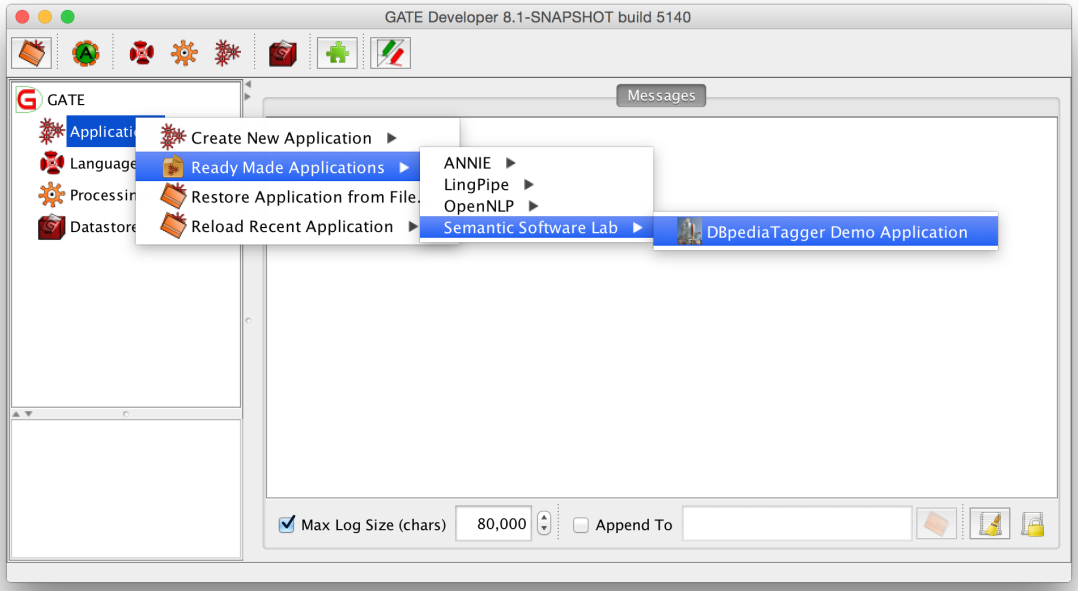

Figure 1.2: Loading the DBpediaTagger Demo Application in GATE

JAPE transducer then filters out any tagged entity that does not fall within the boundary of a noun phrase.

Note that in order to use the DBpediaTagger, you must have an instance of DBpedia Spotlight web service running on a local or remote connection and properly point the DBpediaTagger component to the web service's RESTful endpoint. For more information on how to install and run Spotlight, please refer to the DBpedia Spotlight user guide.[6] Note that the demo pipeline uses a public Spotlight web service by default, but for production use you should change that to a custom endpoint.

1.1.3 Runtime parameters

In order to run the DBpediaTagger component, you can set the following run-time parameters:

confidence the required confidence of the Spotlight tool in disambiguating a term;

endpoint the URL of the *annotate* service on the server where the DBpedia Spotlight web service is running (the default is `http://localhost:2222/rest/annotate`, i.e., a local Spotlight installation);

outputASName output annotation set;

outputAnnotationName name (type) of the annotations generated from the results; and

support is the minimum number of inlinks a DBpedia resource must have to be annotated. The higher the number, the more likely that popular terms are annotated and rare terms are missed in the text.

[6] DBpedia Spotlight User Guide, https://github.com/dbpedia-spotlight/dbpedia-spotlight/wiki/Installation

1.2 Implementation notes

1.2.1 DBpediaTagger

Technically, DBpediaTagger is a wrapper program written in Java based on the GATE framework, which provides a simplified interface to interact with a DBpedia Spotlight web service from within the GATE environment.

The component sends the entire UTF-8 formatted text of a document as a RESTful POST request to Spotlight. It accepts the results in JSON format and translates each JSON object in the response body to a new annotation in GATE. Each entity detected by Spotlight also bears the start and end offsets of its surface form in the text that directly matches its offsets in the GATE document. Since users can choose to define the name of the generated annotations, the DBpediaTagger creates a new annotation of the specified type within the spans defined by the offsets. Figure 1.3 shows an example JSON result that corresponds to the annotations shown in Figure 1.1.

In our current implementation, in order save memory DBpediaTagger only preserves the `URI` and `similarityScore` features of each annotation and discards the rest. If you would like to store additional features in the generated annotations, you can extend the component's main class located in `DBpediaTagger/src/info/semanticsoftware/lodtagger/DBpediaTagger.java` file and modify the following lines:

```
@Override
public void execute() throws ExecutionException {

    String docContent = document.getContent().toString();
    final SpotlightResult result = callSpotlight(docContent);
    final List<SpotlightResource> list = result.getResources();
    AnnotationSet outputAS = document.getAnnotations(outputASName);

    for (SpotlightResource rsrc : list) {
        FeatureMap feats = Factory.newFeatureMap();
        feats.put("URI", rsrc.getURI());
        feats.put("similarityScore", rsrc.getSimilarityScore());

        // copy the line below for each feature and replace FEATURE_NAME with the proper name, e.g. "surfaceForm"
        feat.put(FEATURE_NAME, rsc.getFEATURE_NAME());

        if (rsrc.getSurfaceForm() == null) {
            rsrc.setSurfaceForm("null");
        }
        final long endOffset = rsrc.getOffset() + rsrc.getSurfaceForm().length();
        try {
            outputAS.add(rsrc.getOffset(), endOffset, getOutputAnnotationName(), feats);
        } catch (InvalidOffsetException e) {
            e.printStackTrace();
        }
    }
}
```

1.3 Troubleshooting

How do I know if my local Spotlight is running properly?

If you are using a local installation of Spotlight, you can test its endpoint using your command line. Assuming that the Spotlight server is running on your `http://localhost:2222`, copy and paste the following command in your command line environment:

```
curl -H "Accept: application/json" http://localhost:2222/rest/annotate \
  --data-urlencode "text=Albert Einstein was born in Ulm, Germany." \
  --data "confidence=0.2" \
  --data "support=20"
```

```
{
  "@text": "Albert Einstein was born in Ulm, Germany.",
  "@confidence": "0.2",
  "@support": "0",
  "@types": "",
  "@sparql": "",
  "@policy": "whitelist",
  "Resources":   [
        {
      "@URI": "http://dbpedia.org/resource/Albert_Einstein",
      "@support": "3948",
      "@types": "DBpedia:Agent,Schema:Person,Http://xmlns.com/foaf/0.1/Person,DBpedia:Person,
    DBpedia:Scientist",
      "@surfaceForm": "Albert Einstein",
      "@offset": "0",
      "@similarityScore": "0.9999999999890434",
      "@percentageOfSecondRank": "8.271020058100814E-12"
    },
        {
      "@URI": "http://dbpedia.org/resource/Name_at_birth",
      "@support": "1688",
      "@types": "",
      "@surfaceForm": "born",
      "@offset": "20",
      "@similarityScore": "0.6264053011004147",
      "@percentageOfSecondRank": "0.4805726283431195"
    },
        {
      "@URI": "http://dbpedia.org/resource/Ulm",
      "@support": "1621",
      "@types": "Schema:Place,DBpedia:Place,DBpedia:PopulatedPlace,DBpedia:Settlement,DBpedia
    :Town",
      "@surfaceForm": "Ulm",
      "@offset": "28",
      "@similarityScore": "0.9918617277820427",
      "@percentageOfSecondRank": "0.008205046896717522"
    },
        {
      "@URI": "http://dbpedia.org/resource/Germany",
      "@support": "182228",
      "@types": "Schema:Place,DBpedia:Place,DBpedia:PopulatedPlace,Schema:Country,DBpedia:
    Country",
      "@surfaceForm": "Germany",
      "@offset": "33",
      "@similarityScore": "0.9991828453164097",
      "@percentageOfSecondRank": "4.4933686876209676E-4"
    }
  ]
}
```


Figure 1.3: JSON response from Spotlight for the document shown in Figure 1.1

The result from the endpoint should match the code in Figure 1.3. Note that depending on your Spotlight model, the results might vary in their confidence or similarity scores.

I get "java.net.ConnectException: Connection refused" when running the pipeline.

This exception is thrown when the DBpediaTagger component cannot access the configured Spotlight RESTful endpoint. This means either the endpoint is offline or it in unaccessible (e.g., behind a firewall).

1.4 Changelog

Version 1.1 (07.10.2016)

Added "types" feature to the output annotation.

Version 1.0 (24.07.2015)

Initial public release.