GeoTopicParser

GeoTopicParser

The GeoTopicParser combines a Gazetteer (a lookup dictionary of names/places to latitudes, longitudes) and a Named Entity Recognition (NER) modeling technique that identifies names and places in text to provide a way to geo tag documents and text i.e., to identify places in the text, and then to look up the latitude/longitude pairs for those places.

GeoTopicParser uses Geonames.org, Apache Lucene and Apache OpenNLP to provide its capabilities.

The work is based on this paper:

An Automatic Approach for Discovering and Geocoding Locations in Domain-Specific Web Data

In Proceedings of the IEEE International Conference on Information Reuse and Integration, Pittsburgh, Pennsylvania, USA, July 28-30, 2016 | Read this article

Authors: Chris A. Mattmann, Madhav Sharan

Installing the Lucene Gazetteer

First you will need to download the Lucene Geo Gazetteer project and to install it. You can do so by:

```
$ cd $HOME/src
$ git clone https://github.com/chrismattmann/lucene-geo-gazetteer.git
$ cd lucene-geo-gazetteer
$ mvn install assembly:assembly
$ add $HOME/src/lucene-geo-gazetteer/src/main/bin to your PATH environment variable
```

Once done, you can verify that the installation worked by running the following command:

```
$ lucene-geo-gazetteer --help
usage: lucene-geo-gazetteer
 -b,--build <gazetteer file>           The Path to the Geonames
                                       allCountries.txt
 -c,--count <number of results>        Number of best results to be
                                       returned for one location
 -h,--help                             Print this message.
 -i,--index <directoryPath>            The path to the Lucene index
                                       directory to either create or read
 -json,--json                          Formats output in well defined json
                                       structure
 -s,--search <set of location names>   Location names to search the
                                       Gazetteer for
 -server,--server                      Launches Geo Gazetteer Service
```

You will now need to build a Gazetteer using the Geonames.org dataset. Instructions are provided below. Note that you will need least 1.2 GB disk space for building Lucene Index for the Gazetteer.

```
$ cd $HOME/src/lucene-geo-gazetteer
$ curl -O http://download.geonames.org/export/dump/allCountries.zip
$ unzip allCountries.zip
$ lucene-geo-gazetteer -i geoIndex -b allCountries.txt
```

You can verify that the Gazetteer build worked by searching e.g., for Pasadena, and/or Texas:

```
$ lucene-geo-gazetteer -s Pasadena Texas -json
{
    "Pasadena": [
        {
            "admin1Code": "CA",
            "admin2Code": "037",
            "countryCode": "US",
            "latitude": 34.14778,
            "longitude": -118.14452,
            "name": "Pasadena"
        }
    ],
    "Texas": [
        {
            "admin1Code": "TX",
            "admin2Code": "",
            "countryCode": "US",
            "latitude": 31.25044,
            "longitude": -99.25061,
            "name": "Texas"
        }
    ]
}
```

Now you need to start REST service of lucene-geo-gazetteer. Tika uses this service internally

```
$ lucene-geo-gazetteer -server
```

You can verify that the REST API is responding by searching e.g., for Pasadena, and/or Texas:

```
$ curl "http://localhost:8765/api/search?s=Pasadena&s=Texas"
{
    "Pasadena": [
        {
            "admin1Code": "CA",
            "admin2Code": "037",
            "countryCode": "US",
            "latitude": 34.14778,
            "longitude": -118.14452,
            "name": "Pasadena"
        }
    ],
    "Texas": [
        {
            "admin1Code": "TX",
            "admin2Code": "",
            "countryCode": "US",
            "latitude": 31.25044,
            "longitude": -99.25061,
            "name": "Texas"
        }
    ]
}
```

Note that we used the convenience script `lucene-geo-gazetteer` which assumes that you created an indexed named geoIndex in the $HOME/src /lucene-geo-gazetter/geoIndex directory. We could have also used the pure Java command line to search. The return from the Gazetteer is a JSON List of Object structures in which the structure is a key->Object List map. The key is the location name given and the Object List is a list of most popular location objects in the Gazetteer for that name.

Installing and downloading an NER model

The next thing you'll need is a Named Entity Recognition model for places. The GeoTopicParser uses Apache OpenNLP and with its 1.5 version, OpenNLP provides already trained models for location names in text data. You can download the en-ner-location.bin file already pre-trained by the OpenNLP folks. One thing to note is that OpenNLP's default name finder is not accurate, so building your own NER location model is highly recommended. In this case, please follow these instructions.

The model needs to be placed on the classpath for your Tika installation in the following directory:

`org/apache/tika/parser/geo/`

The following instructions show how to download the model and place it on the right path:

```
$ mkdir $HOME/src/location-ner-model && cd $HOME/src/location-ner-model
$ curl -O https://opennlp.sourceforge.net/models-1.5/en-ner-location.bin
$ mkdir -p org/apache/tika/parser/geo/
$ mv en-ner-location.bin org/apache/tika/parser/geo/
```

Test out the GeoTopicParser

Now you can run Tika and try out the GeoTopicParser. At the moment since it's a Parser and not a Content-Handler (hopefully will develop it later), the parser is mapped to the MIME type application/geotopic which is a sub-class of text/plain. So, there are two steps to try the parser out now.

1. Create a .geot file, you can use this sample file from the NSF Polar data contributed to TREC. 2. Tell Tika about the application/geotopic MIME type. You can download this file and place it on the classpath in the `org/apache/tika/mime` directory, e.g., by doing:

```
$ mkdir $HOME/src/geotopic-mime && cd $HOME/src/geotopic-mime
$ mkdir -p org/apache/tika/mime
$ curl -O https://raw.githubusercontent.com/chrismattmann/geotopicparser-utils/master/mime/org/apache
/tika/mime/custom-mimetypes.xml
$ mv custom-mimetypes.xml org/apache/tika/mime
```

With those files in place, let's use the GeoTopicParser using Tika-App:

```
$ java -classpath tika-app/target/tika-app-<LATEST-VERSION>-SNAPSHOT.jar:tika-parsers/tika-parsers-ml/tika-
parser-nlp-package/target/tika-parser-nlp-package-<LATEST-VERSION>-SNAPSHOT.jar:$HOME/src/location-ner-
model:$HOME/src/geotopic-mime org.apache.tika.cli.TikaCLI -m polar.geot
```

This should output:

```
Content-Length: 881
Content-Type: application/geotopic
Geographic_LATITUDE: 27.33931
Geographic_LONGITUDE: -108.60288
Geographic_NAME: China
Optional_LATITUDE1: 39.76
Optional_LONGITUDE1: -98.5
Optional_NAME1: United States
X-Parsed-By: org.apache.tika.parser.DefaultParser
X-Parsed-By: org.apache.tika.parser.geo.topic.GeoParser
resourceName: polar.geot
```

The output will output 3-tuples of {`Name, Latitude, Longitude`}. The *best* match for the location is the one that occurs most frequently in the text, and that is provided as `Geographic_NAME`, along with its corresponding `Geographic_LATITUDE` and `Geographic_LONGITUDE`. Places also identified as entities by the NER model in the provide text are also listed as `Optional_NAME*N*`, e.g., `Optional_NAME1` for the 1st alternative location identified and its corresponding `Optional_LATITUDE1` and `Optional_LONGITUDE1`.

Will this work from Tika Server?

It sure will! When you start Tika Server, make sure that the NER model file and the custom MIME type are on your classpath, and that the lucene-geo-gazetteer is on the `$PATH` where Tika-Server is started, and you can post all the .geot files that you'd like and Tika-Server will happily call the GeoTopicPar ser to provide you location information.

First, start up the Tika server with your NER model and .geot MIME type definition on the classpath:

```
java -classpath $HOME/src/location-ner-model:$HOME/src/geotopic-mime:tika-server/tika-server-standard/target
/tika-server-standard-<LATEST-VERSION>-SNAPSHOT.jar:tika-parsers/tika-parsers-ml/tika-parser-nlp-package/target
/tika-parser-nlp-package-<LATEST-VERSION>-SNAPSHOT.jar org.apache.tika.server.core.TikaServerCli
```

Then, try calling the `/rmeta` service to get the returned metadata:

```
curl -T $HOME/src/geotopicparser-utils/geotopics/polar.geot -H "Content-Disposition: attachment; filename=polar.
geot" http://localhost:9998/rmeta
```

And then look for it to return the following, that's it!

```
[
   {
      "Content-Type":"application/geotopic",
      "Geographic_LATITUDE":"39.76",
      "Geographic_LONGITUDE":"-98.5",
      "Geographic_NAME":"United States",
      "Optional_LATITUDE1":"27.33931",
      "Optional_LONGITUDE1":"-108.60288",
      "Optional_NAME1":"China",
      "X-Parsed-By":[
         "org.apache.tika.parser.DefaultParser",
         "org.apache.tika.parser.geo.topic.GeoParser"
      ],
      "X-TIKA:parse_time_millis":"1634",
      "resourceName":"polar.geot"
   }
]
```