Named Entity Recognition (NER) with Tika

Named Entity Recognition is supported in tika-parsers, introduced in Tika-1787. This page describes the steps required to configure and activate the Named Entity Parser.

Activate Named Entity Parser

Before moving ahead to configure NER implementations, org.apache.tika.parser.ner.NamedEntityParser, the parser responsible for handling the name recognition task needs to be enabled. This can be done with Tika Config XML file, as follows:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<properties>
  <parsers>
    <parser class="org.apache.tika.parser.ner.NamedEntityParser">
      <mime>text/plain</mime>
      <mime>text/html</mime>
      <mime>application/xhtml+xml</mime>
    </parser>
  </parsers>
</properties>
```

This configuration has to be supplied in the later phases, so store it as `tika-config.xml`.

Note: The parser does not restrict mimetypes, it uses Tika's auto detect parser to read text content from non-text streams.

Using Apache OpenNLP NER

The NE Parser is configured to use an implementation based on Apache OpenNLP. However, the NER models need to be added to the Tika's classpath to make this work.

The following table shows types of entities and the paths to place the model file.

<table>
<thead>
<tr>
<th>Entity Type</th>
<th>Path for model</th>
<th>URL to get</th>
</tr>
</thead>
<tbody>
<tr>
<td>PERSON</td>
<td>org/apache/tika/parser/ner/opennlp/ner-person.bin</td>
<td><a href="http://opennlp.sourceforge.net/models-1.5/en-ner-person.bin">http://opennlp.sourceforge.net/models-1.5/en-ner-person.bin</a></td>
</tr>
<tr>
<td>LOCATION</td>
<td>org/apache/tika/parser/ner/opennlp/ner-location.bin</td>
<td><a href="http://opennlp.sourceforge.net/models-1.5/en-ner-location.bin">http://opennlp.sourceforge.net/models-1.5/en-ner-location.bin</a></td>
</tr>
<tr>
<td>ORGANIZATION</td>
<td>org/apache/tika/parser/ner/opennlp/ner-organization.bin</td>
<td><a href="http://opennlp.sourceforge.net/models-1.5/en-ner-organization.bin">http://opennlp.sourceforge.net/models-1.5/en-ner-organization.bin</a></td>
</tr>
<tr>
<td>DATE</td>
<td>org/apache/tika/parser/ner/opennlp/ner-date.bin</td>
<td><a href="http://opennlp.sourceforge.net/models-1.5/en-ner-date.bin">http://opennlp.sourceforge.net/models-1.5/en-ner-date.bin</a></td>
</tr>
<tr>
<td>MONEY</td>
<td>org/apache/tika/parser/ner/opennlp/ner-money.bin</td>
<td><a href="http://opennlp.sourceforge.net/models-1.5/en-ner-money.bin">http://opennlp.sourceforge.net/models-1.5/en-ner-money.bin</a></td>
</tr>
</tbody>
</table>

Notes:

1. You can use any combination of the models. If you are interested in only the LOCATION names, then skip other NER models save LOCATION.
2. NER Models for other languages are also available http://opennlp.sourceforge.net/models-1.5/. If you choose to use different language, use those URLs in the below script.
# Create a directory for keeping all the models.
# Choose any convenient path but make sure to use absolute path
export NER_RES=$HOME/tika/tika-ner-resources
mkdir -p $NER_RES
cd $NER_RES

PATH_PREFIX="$NER_RES/org/apache/tika/parser/ner/opennlp"
URL_PREFIX="http://opennlp.sourceforge.net/models-1.5"

mkdir -p $PATH_PREFIX

# using three entity types from the above table for demonstration
wget "$URL_PREFIX/en-ner-person.bin" -O $PATH_PREFIX/ner-person.bin
wget "$URL_PREFIX/en-ner-location.bin" -O $PATH_PREFIX/ner-location.bin
wget "$URL_PREFIX/en-ner-organization.bin" -O $PATH_PREFIX/ner-organization.bin

export TIKA_APP={your/path/to/tika-app}/target/tika-app-1.12-SNAPSHOT.jar

# Are there any metadata keys starting with "NER_"?

## Using Stanford CoreNLP NER

The 'org.apache.tika.parser.ner.corenlp.CoreNLPNERecogniser' class provides runtime bindings to Stanford CoreNLP CRF classifiers for named entity recognition.

The following steps are necessary to use this NER implementation:

- Add Core NLP library and its dependencies to classpath
- Add models to class path
- Set NER Implementation to CoreNLP

**NOTE:** The latest release of Stanford CoreNLP requires JDK8.

### Tika + CoreNLP in action

cd /$HOME/src
git clone https://github.com/thammegowda/tika-ner-corenlp.git
cd tika-ner-corenlp
mvn clean compile package assembly:single -PtikaAddon

# this should produce target/tika-ner-corenlp-addon-*-jar-with-dependencies.jar
export CORE_NLP_JAR=`find $PWD/target/tika-ner-corenlp-addon-*-jar-with-dependencies.jar`
export TIKA_APP={your/path/to/tika-app}/target/tika-app-1.12-SNAPSHOT.jar


# Observe metadata keys starting with NER_

# To use 3class NER model (Default is 7 class model)


The CoreNLP CRF classifier recognised the following from the text content of http://www.hawking.org.uk page:
Using Regular Expressions

The `org.apache.tika.parser.ner.regex.RegexNERecogniser` provides an implementation based on Regular expressions. The following steps are required to use this implementation:

- Configure regular expressions in `org/apache/tika/parser/ner/regex/ner-regex.txt`
- Set System property `ner.impl.class` to `org.apache.tika.parser.ner.regex.RegexNERecogniser`

Tika + RegexNER in action

```bash
# Create a regex file and add it to classpath
export NER_RES=$HOME/tika/tika-ner-resources
cd $NER_RES
mkdir -p org/apache/tika/parser/ner/regex/
echo "PHONENUMBER=(\+\d{1,2}\s?)?\(?\d{3}\)?\s*-\s*-?\d{3}\s*-\s*-?\d{4}\)" > org/apache/tika/parser/ner/regex/ner-regex.txt
echo "EMAIL=(\[a-zA-Z0-9.!#$%&'*+/=?^_`{|}~-.]+@[a-zA-Z0-9.(?:[a-zA-Z0-9-]{0,61}[a-zA-Z0-9])\?\?::\[a-zA-Z0-9-]{0,61}[a-zA-Z0-9]]\)
" >> org/apache/tika/parser/ner/regex/ner-regex.txt
export TIKA_APP={your/path/to/tika-app}/target/tika-app-1.12-SNAPSHOT.jar
java -Dner.impl.class=org.apache.tika.parser.ner.regex.RegexNERecogniser \
--config=tika-config.xml --m http://www.cs.usc.edu/faculty_staff/faculty

# Observe values of keys NER_PHONE_NUMBER and NER_EMAIL
```

Creating a custom NER

- Create a class and implement `org.apache.tika.parser.ner.NERecogniser`
- Set class name as value to system property `ner.impl.class` similar to Regex or CoreNLP
Chaining all the above at once

Multiple class names can be provided by setting the system property `ner.impl.class` to a comma separated class names

Example: `-Dner.impl.class = org.apache.tika.parser.ner.opennlp.OpenNLPNERecogniser,org.apache.tika.parser.ner.regex.RegexNERecogniser`