TikaAndVision

Tika and Computer Vision

This page describes how to make use of Object (Visual) Recognition capability of Apache Tika. TIKA-1993 introduced a new parser to perform object recognition on images. Visit TIKA-1993 issue on Jira or pull request on Github to read the related conversation. The model was updated from Inception V3 to Inception V4 with TIKA-2306 (Pull request on Github). Continue reading to get Tika up and running for object recognition.

Currently, Tika utilises Inception-V4 model from Tensorflow for recognizing objects in the JPEG images.

Tika and Tensorflow Image Recognition

Tika has two different ways of bindings to Tensorflow:

1. Using Commandline Invocation -- Recommended for quick testing, not for production use
2. Using REST API -- Recommended for production use

1. Tensorflow Using Commandline Invocation

Pros of this approach:

This parser is easy to setup and test

Cons:

Very inefficient/slow as it loads and unloads model for every parse call

Step 1. Install the dependencies

To install tensorflow, follow the instructions on the official site here for your environment. Unless you know what you are doing, you are recommended to follow pip installation.

Then clone the repository tensorflow/models or download the zip file.

```
git clone https://github.com/tensorflow/models.git
```

Add 'models/slim' folder to the environment variable, PYTHONPATH.

```
$ export PYTHONPATH="$PYTHONPATH:/path/to/models/slim"
```

To test the readiness of your environment :

```
$ python -c 'import tensorflow, numpy, dataset; print("OK")'
```

If the above command prints the message "OK", then the requirements are satisfied.

Step 2. Create a Tika-Config XML to enable Tensorflow parser.

A sample config can be found in Tika source code at tika-parsers/src/test/resources/org/apache/tika/parser/recognition/tika-config-tflow.xml

Here is an example:

Toggle line numbers

```
<properties>
    <parsers>
        <parser class="org.apache.tika.parser.recognition.ObjectRecognitionParser">
            <mime>image/jpeg</mime>
            <params>
                <param name="topN" type="int">2</param>
                <param name="minConfidence" type="double">0.015</param>
                <param name="class" type="string">org.apache.tika.parser.recognition.tf.
TensorflowImageRecParser</param>
            </params>
        </parser>
    </parsers>
</properties>
```

Description of parameters :

Param Name	Type	Meaning	Range	Example
topN	int	Number of object names to output	a non-zero positive integer	1 to receive top 1 object name
minConfidence	double	Minimum confidence required to output the name of detected objects	[0.0 to 1.0] inclusive	0.9 for outputting object names iff at least 90% confident
class	string	Class that implements object recognition functionality	constant string	org.apache.tika.parser.recognition.tf.TensorflowImageRecParser

Step 3: Demo

To use the vision capability via Tensorflow, just supply the above configuration to Tika.

For example, to use in Tika App (Assuming you have *tika-app* JAR and it is ready to run):

```
      $ java -jar tika-app/target/tika-app-1.15-SNAPSHOT.jar \
         --config=tika-parsers/src/test/resources/org/apache/tika/parser/recognition/tika-config-tflow.xml \
         https://upload.wikimedia.org/wikipedia/commons/f/f6/Working_Dogs%
2C_Handlers_Share_Special_Bond_DVIDS124942.jpg
```

The input image is:

blocked URL

And, the top 2 detections are:

Toggle line numbers

```
...
<meta name="OBJECT" content="German shepherd, German shepherd dog, German police dog, alsatian (0.78435)"/>
<meta name="OBJECT" content="military uniform (0.06694)"/>
...
```

2. Tensorflow Using REST Server

This is the recommended way for utilizing visual recognition capability of Tika. This approach uses Tensorflow over REST API. To get this working, we are going to start a python flask based REST API server and tell tika to connect to it. All these dependencies and setup complexities are isolated in docker image.

Requirements :

Docker -- Visit Docker.com and install latest version of Docker. (Note: tested on docker v17.03.1)

Step 1. Setup REST Server

You can either start the REST server in an isolated docker container or natively on the host that runs tensorflow.

a. Using docker (Recommended)

Toggle line numbers

```
1 git clone https://github.com/USCDataScience/tika-dockers.git && cd tika-dockers
2 docker build -f InceptionRestDockerfile -t uscdatascience/inception-rest-tika .
3 docker run -p 8764:8764 -it uscdatascience/inception-rest-tika
```

Once it is done, test the setup by visiting http://localhost:8764/inception/v4/classify/image?topn=2&min_confidence=0.03&url=https://upload.wikimedia.org/wikipedia/commons/f/f6/Working_Dogs%2C_Handlers_Share_Special_Bond_DVIDS124942.jpg in your web browser.

Sample output from API:

```
{
   "confidence":[
      0.7843596339225769,
      0.06694009155035019
   ],
   "classnames":[
      "German shepherd, German shepherd dog, German police dog, alsatian",
      "military uniform"
   ],
   "classids":[
      236,
      653
   ],
   "time":{
      "read":7403,
      "units":"ms",
      "classification":470
   }
}
```

Note: MAC USERS:

> If you are using an older version, say, 'Docker toolbox' instead of the newer 'Docker for Mac',

you need to add port forwarding rules in your Virtual Box default machine.

1. Open the Virtual Box Manager.
2. Select your Docker Machine Virtual Box image.
3. Open Settings -> Network -> Advanced -> Port Forwarding.
4. Add an appname,Host IP 127.0.0.1 and set both ports to 8764.

b. Without Using docker

If you chose to setup REST server without a docker container, you are free to manually install all the required tools specified in the docker file.

Note: docker file has setup instructions for Ubuntu, you will have to transform those commands for your environment.

Toggle line numbers

```
1    python tika-parsers/src/main/resources/org/apache/tika/parser/recognition/tf/inceptionapi.py  --port 8764
```

Step 2. Create a Tika-Config XML to enable Tensorflow parser.

> A sample config can be found in Tika source code at tika-parsers/src/test/resources/org/apache/tika/parser/recognition/tika-config-tflow-rest.xml

Here is an example:

```
<properties>
    <parsers>
        <parser class="org.apache.tika.parser.recognition.ObjectRecognitionParser">
            <mime>image/jpeg</mime>
            <params>
                <param name="topN" type="int">2</param>
                <param name="minConfidence" type="double">0.015</param>
                <param name="class" type="string">org.apache.tika.parser.recognition.tf.TensorflowRESTRecogniser<
/param>
            </params>
        </parser>
    </parsers>
</properties>
```

Description of parameters :

Param Name	Type	Meaning	Range	Example
topN	int	Number of object names to output	a non-zero positive integer	1 to receive top 1 object name
minConfidence	double	Minimum confidence required to output the name of detected objects	[0.0 to 1.0] inclusive	0.9 for outputting object names iff at least 90% confident
class	string	Name of class that Implements Object recognition Contract	constant string	org.apache.tika.parser.recognition.tf.TensorflowRESTRecogniser
healthUri	uri	HTTP URL to check availability of API service	any HTTP URL that gets 200 status code when available	http://localhost:8764/inception/v4/ping
apiUri	uri	HTTP URL to POST image data	any HTTP URL that returns data in the JSON format as shown in the sample API output	http://localhost:8764/inception/v4/classify/image?topk=10

Step 3. Demo

This demo is same as the Commandline Invocation approach, but this is faster and efficient

```
   $ java -jar tika-app/target/tika-app-1.15-SNAPSHOT.jar \
    --config=tika-parsers/src/test/resources/org/apache/tika/parser/recognition/tika-config-tflow-rest.xml \
    https://upload.wikimedia.org/wikipedia/commons/f/f6/Working_Dogs%2C_Handlers_Share_Special_Bond_DVIDS124942.
jpg
```

The input image is:

blocked URL

And, the top 2 detections are:

Toggle line numbers

```
1 ...
2 <meta name="OBJECT" content="German shepherd, German shepherd dog, German police dog, alsatian (0.78435)"/>
3 <meta name="OBJECT" content="military uniform (0.06694)"/>
4 ...
```

Changing the default topN, API port or URL

To change the defaults, update the parameters in config XML file accordingly

Here is an example scenario:

Run REST API on port 3030, and get top 4 object names if the confidence is above 10%. You may also change host to something else than 'localhost' if required.

Example Config File

```
<properties>
    <parsers>
        <parser class="org.apache.tika.parser.recognition.ObjectRecognitionParser">
            <mime>image/jpeg</mime>
            <params>
                <param name="topN" type="int">4</param>
                <param name="minConfidence" type="double">0.1</param>
                <param name="class" type="string">org.apache.tika.parser.recognition.tf.TensorflowRESTRecogniser<
/param>
                <param name="healthUri" type="uri">http://localhost:3030/inception/v4/ping</param>
                <param name="apiUri" type="uri">http://localhost:3030/inception/v4/classify/image?topk=4</param>
            </params>
        </parser>
    </parsers>
</properties>
```

To Start the service on port 3030:

Using Docker:

```
docker run -it -p 3030:8764 uscdatascience/inception-rest-tika
```

Without Using Docker:

python tika-parsers/src/main/resources/org/apache/tika/parser/recognition/tf/inceptionapi.py --port 3030

Questions / Suggestions / Improvements / Feedback ?

1. If it was useful, let us know on twitter by mentioning @ApacheTika
2. If you have questions, let us know by using Mailing Lists
3. If you find any bugs, use Jira to report them