Poio Documentation

Release 0.1.0

Peter Bouda, Ricardo Filipe

January 17, 2014

Contents

The Poio project develops language technologies to support communication in lesser-used and under-resourced languages on and with electronic devices. Within the Poio project we develop text input services with text prediction and transliteration for mobile devices and desktop users to allow conversation in hundreds of languages between individuals and in online communities. Poio started as a language revitalization project at the Interdisciplinary Centre for Social and Language Documentation in Minde/Portugal, a non-profit organization dedicated to the documentation and preservation of linguistic heritage.

You can try the text prediction on the official website of Poio:

http://www.poio.eu

Poio consists of several open source projects to process and manage language data, to extract corpora from diverse data sources and to calculate language models for the online tools. You can find a list of all Poio projects on the website of the Interdisciplinary Centre for Social and Language Documentation:

http://media.cidles.eu/poio/

CHAPTER 1

Contents

1.1 Introduction

There are currently around 7.000 languages spoken all over the world - but each two weeks a language dies. In general we see a strong tendency to learn and use only the major languages such as English, German, French, etc., especially in electronic communication. This is partly caused by the lack of hardware (e.g. keyboards) and software (for transliteration, text completion, etc.) for lesser-used languages, which constrain the natural usage of people's own language in many tasks. Our goal is to turn that process in exactly the other direction: every successful technology can also be used to teach, revitalize and therefore boost the use of regional languages. This technology should also assist the renewal of local languages and cultures by allowing people to actively teach, learn, extend, and spread their language in their community. Our aim is to give people the ability to use their mother tongue in everyday electronic communication, no matter where they are and whatever language they speak.

The Poio project develops language technologies to support communication in lesser-used and under-resourced languages on and with electronic devices. Within the Poio project we develop text input services with text prediction and transliteration for mobile devices and desktop users to allow conversation in hundreds of languages between individuals and in online communities. Poio started as a language revitalization project at the Interdisciplinary Centre for Social and Language Documentation in Minde/Portugal, a non-profit organization dedicated to the documentation and preservation of linguistic heritage.

1.2 Newsletter

If you want to get regular updates on the progress of the Poio project please subscribe to our newsletter:

Subscribe to weekly updates on the Poio project

1.3 Become a tester

If you want to become a beta tester for new functionality and tools then please subsribe to our newsletter for testers, with regular updates on new features before they go public:

Subscribe as a beta tester

1.4 How to use the corpus files

The corpus files of Poio are mostly collected and extracted from the Wikipedias in the different languages. For this, we use the official Wikipedia dumps and convert them to ISO 24612 (GrAF-XML). Peter Bouda wrote a blog post about the whole process:

http://www.peterbouda.eu/parsing-wikipedia-dumps-and-conversion-to-iso-24612-graf-xml.html

This data is then fed into the pressagio library to build a language model for the text prediction. A follow-up blog post about this workflows is available here:

http://www.peterbouda.eu/pressagio-a-predictive-text-system-in-python.html

To parse the content of the GrAF-XML files you can use the Python library graf-python. First, you have to download one the corpus files. See *api-corpusfiles* on how to use the official API of Poio to get a list of all corpus files for a given language.

For example, the Wikipedia corpus for Bavarian is available for download at:

http://www.poio.eu/static/corpus/barwiki.zip

Next, extract the example files. Here is an example code snippet how the individual documents can be read into a Python list:

```
import graf

gp = graf.GraphParser()
g = gp.parse("barwiki-20131229.txt")
documents = list()
for n in g.nodes:
    if n.id.startswith("doc..") and len(n.links) > 0 and len(n.links[0]) > 0:
        doc = txt[n.links[0][0].start:n.links[0][0].end]
        documents.append(doc)
```

This will store all the documents as strings in a list `documents`.

1.5 Web API of Poio

1.5.1 Text Prediction

The text prediction functionality of Poio is accessible via an API on mashape. We use any revenue from mashape to support the development of the Poio project. To use any of the functionality of our API you have to create an account on mashape. Find the documentation of the API and more information about pricing here:

https://www.mashape.com/pbouda/poio

Or you can deploy Poio on your own server, see *Deploy and develop Poio* for more information.

1.5.2 Supported Languages

This function checks for all the supported languages and returns them as ISO 639-3 codes.

URL:

`<ROOT>/api/languages`

Parameters

This API method takes no parameters.

Return

- languages_list (json)

The data is returned as JSON in the HTTP response. The returned data is a list (array) of all the supported languages as ISO 639-3 codes.

For example:

http://www.poio.eu/api/languages

1.5.3 Corpus Files

This function looks for all the avaible corpus files for a given language and returns a list URLs to those files.

URL:

```
<ROOT>/api/corpus
```

Parameters

- iso (string)

Pass the ISO 639-3 code of the language for that you want to get a list of corpus files as a GET paramter `iso` in the HTTP request:

```
<ROOT>/api/corpus?iso=<iso>
```

Return

- files (json)

The data is returned as JSON in the HTTP response. The returned data is a list (array) of all the paths for all the avaible corpus files for the given language.

For example:

http://www.poio.eu/api/corpus?iso=bar

For more information on how to use the corpus files, see *How to use the corpus files*.

1.6 Deploy and develop Poio

1.6.1 Poio Website

The repository is here:

https://github.com/cidles/poio-site

This will install all requirements and prepare the server (*Flask webapp*) for launch. The Poio website used Buildout to install all the necessary packages. The Poio website comes with a bootstrap script, so you don't have to install anything except Python.

Initialize your enviroment

Start by bootstraping the buildout environment:

```
$ python bootstrap.py
```

Next you have to install all dependencies using the buildout script:

```
$ sudo bin/buildout
```

Get language data from server

You have to download all the language data from our Amazon server (this may take a while):

```
$ python get_corpus_data.py
```

Run the tests

Before starting the server you should run our tests to ensure that everything is working properly:

```
$ bin/test
```

Start the server in development mode

Finally you can start the server:

```
$ bin/flask-ctl debug fg
```

Example deployment on Ubuntu

For deployment we use gunicorn, supervisor and nginx. First, install *nginx* and *supervisor* via `apt-get`:

```
# apt-get install nginx
# apt-get install supervisor
```

The example configuration that comes with the *poio-site* project and the following steps assume that the code of the project is in the directory `/home/sites/poio`. Please change this path in all configurations options if you installed the code somewhere else.

Gunicorn is already installed in the buildout environment if you followed the instruction in *Initialize your enviroment*. An example gunicorn config file gunicorn.conf.py is in the main folder of the *poio-site* project. You may try to start the gunicorn server manually:

```
$ bin/gunicorn main:app -c gunicorn.conf.py -p gunicorn.pid
```

The supervisor daemon is then responsible to start the gunicorn server and keep it running, i.e. restarting it whenever it stopped or crashed. For the Poio website we need a supervisor configuration script, which we store in `/etc/supervisor/conf.d/poio.conf`. The content is:

```
[group:poio]
programs=poio

[program:poio]
command=/home/sites/poio/bin/gunicorn main:app -c gunicorn.conf.py -p gunicorn.pid
directory=/home/sites/poio
autostart=true
autorestart=true
redirect_stderr=true
```

You now have to tell supervisor that we have a new application. This will also automatically start the gunicorn process:

```
# supervisorctl update
```

Restarting the website is also done via `supervisorctl`:

```
# supervisorctl restart poio
```

The last step is now to tell nginx tp proxy all requests to the gunicorn process. This will make the website available to the public. An example configuration file for nginx is:

```
upstream poio {
  server 127.0.0.1:8200;
}

server {
  server_name             yourdomain.com;
  listen                  80;
  keepalive_timeout       70;

  access_log              /var/log/nginx/access-poio.eu.log;
  error_log               /var/log/nginx/error-poio.eu.log;

  location / {
    proxy_redirect      off;
    proxy_set_header    Host                    $host;
    proxy_pass          http://poio;
  }
}
```

We store the configuration file in `/etc/nginx/sites-available/www.poio.eu`. To activate the site you still have to create a symbolic link of the file in the directory `/etc/nginx/sites-enabled`:

```
# ln -s /etc/nginx/sites-available/www.poio.eu /etc/nginx/sites-enabled/
```

Then restart *nginx*:

```
# /etc/init.d/nginx restart
```

That's it. You should now have the Poio website deployed and running on your server.

Develop with PyCharm

- Start by creating a new project with the following settings:
  - Project name: `Poio Site`
  - Location: `<PATH_TO>/poio-site/src/main/`
  - Project type: `Flask Project`

  - Interpreter: `Python 2.7`
- After you press `Ok` PyCharm will prompt if you want to create a project from existing sources, press Yes.
- In order to run the server from PyCharm you need to add a new confguration for the server, to do this:
  - On the menu bar go to `Run` and open `Edit Configurations...`;
  - Press the + sign and from the dropdown menu choose `Python`.
- Fill in the new confifuration with the following settings and press `Ok`:
  - Name: `Poio Site Server`
  - Script: `bin/flask-ctl`
  - Script parameters: `debug fg`
  - Working directory: `<PATH_TO>/poio-site/`

Now every time you want to start the server make sure that the selected configuration on the menu bar is `Poio Site Server` and just press `Run` (play button).

1.6.2 Poio Corpus

The repository is here:

https://github.com/cidles/poio-corpus

Install the requirements for Poio Corpus

You can find a list of all requirements in REQUIREMENTS.txt.

This documentation is for Linux/Ubuntu only.

You need to install the following packages with *apt-get*:

- python-lxml
- python-numpy
- python-scipy

For example:

```
$ sudo apt-get install python-lxml
```

You need to install the following packages with easy_install or pip:

- requests
- beautifulsoup
- graf-python
- poio-api
- rdflib
- cython
- sparsesvd
- regex
- s3cmd

For example:

```
$ sudo easy_install requests
```

In addition, some packages must be downloaded from github:

- pressagio: https://github.com/cidles/pressagio

To install the packages run:

```
$ sudo python setup.py install
```

1.7 Credits

The development of Poio would not have been possible without Open Source Software and Open Data. We use the following services and software to build Poio and provide our services:

- Glottlog: Glottolog is "comprehensive reference information for the world's languages, especially the lesser known languages". Poio uses the RDF resources of Glottolog to get languages names and geo information for our map.
- Wikipedia: We build our language models for the text prediction based on the Wikipedia dumps of the different languages, among others.
- Flask: The Poio website is powered by Flask. Flask is an excellent web application (micro)framework written in Python.
- jQuery Mobile Icon Pack: The SMS and E-Mail button of our mobile website uses buttons from this great icon pack for jQuery Mobile.

CHAPTER 2

Indices and tables

- *genindex*
- *search*