$\textbf{django}_{o}cr_{s}erver$

Release 1.1

Oct 11, 2019

Contents:

CHAPTER 1

Introduction

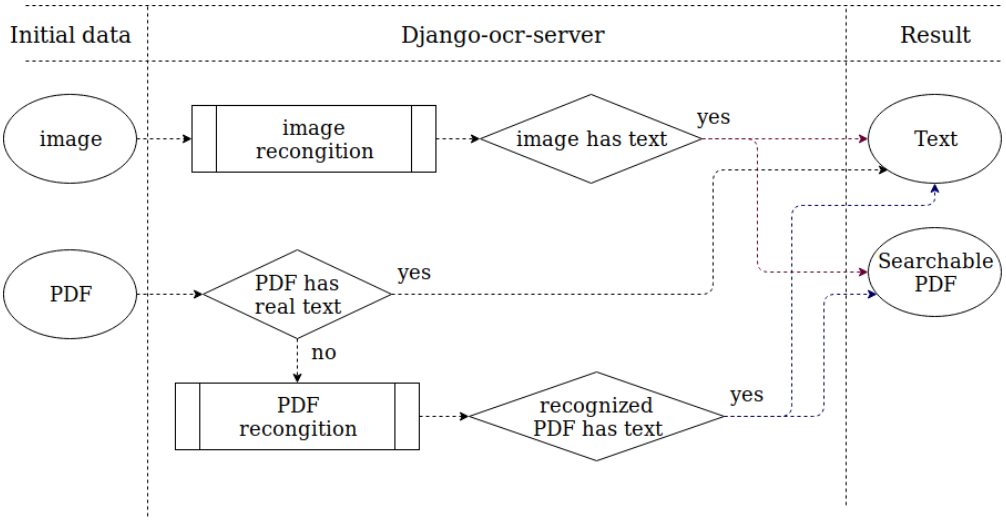
Django-ocr-server lets you recognize images and PDF. It is using tesseract for this. https://github.com/tesseract-ocr/tesseract

Django-ocr-server saves the result in the database. To prevent repeated recognition of the same file, it also saves the hash sum of the uploaded file. Therefore, when reloading an already existing file, the result returns immediately, bypassing the recognition process, which significantly reduces the load on the server.

If as a result of recognition a non-empty text is received, a searchable PDF is created.

For the searchable PDF is calculated hash sum too. Therefore, if you upload the created by Django-ocr-server searchable pdf to the server back, then this file will not be recognized, but the result will be immediately returned.

The server can process not only images, but PDF. At the same time, he analyzes, if the PDF already contains real text, this text will be used and the file will not be recognized, which reduces the load on the server and improves the quality of the output.



Storage of downloaded files and created searchable PDFs can be disabled in the settings.

For uploaded files and created searchable PDFs, and the processing results whole in the settings you can specify the lifetime after which the data will be automatically deleted.

To interact with Django-ocr-server you can use API or the admin interface.

CHAPTER 2

Installation

2.1 Linux Mint 19 (Ubuntu bionic)

Installing packages

$sudo apt install g++ # need to build pdftotext
$sudo apt install libpoppler-cpp-dev # need to buid pdftotext

Installing tesseract

$sudo apt install tesseract-ocr
$sudo apt install tesseract-ocr-rus # install languages you want

Installing python3.7

$sudo apt install python3.7
$sudo apt install python3.7-dev

Installing pip $sudo apt install python-pip

Installing virtualenv

$pip install –user virtualenv
$echo 'PATH=~/.local/bin:$PATH' >> ~/.bashrc
$source ~/.bashrc

Installing virtualenvwrapper

$pip install –user setuptools
$pip install –user wheel
$pip install –user virtualenvwrapper
$echo 'source ~/.local/bin/virtualenvwrapper.sh' >> ~/.bashrc
$source ~/.bashrc

Creating virtualenv for django_ocr_server $mkvirtualenv django_ocr_server -p /usr/bin/python3.7

Installing django-ocr-server (on virtualenv django_ocr_server). It installs Django as a dependency $pip install django-ocr-server

Create your Django project (on virtualenv django_ocr_server) $django-admin startproject ocr_server

Go to project directory $cd ocr_server

Edit ocr_server/settings.py Add applications to INSTALLED_APPS

Edit ocr_server/urls.py

Perform migrations (on virtualenv django_ocr_server) $python manage.py migrate

Create superuser (on virtualenv django_ocr_server) $python manage.py createsuperuser

Run server (on virtualenv django_ocr_server), than visit http://localhost:8000/ $python manage.py runserver

2.2 Linux Mint 19 (Ubuntu bionic) automatic installation

Clone django_ocr_server from github $git clone https://github.com/shmakovpn/django_ocr_server.git

Run the installation script using sudo $sudo {your_path}/django_ocr_server/install_ubuntu.sh

The script creates OS user named 'django_ocr_server', installs all needed packages. Creates the virtual environment. It installs django_ocr_server (from PyPI by default, but you can create the package from cloned repository, see the topic 'Creation a distribution package' how to do this). Then it creates the django project named 'ocr_server' in the home directory of 'django_ocr_server' OS user. After the script changes settings.py and urls.py is placed in ~django_ocr_server/ocr_server/ocr_server/. Finally it applies migrations and creates the superuser named 'admin' with the same password 'admin'.

Run server under OS user django_ocr_server, then change 'admin' password in the http://localhost:your_port/admi

```
$sudo su
$su django_ocr_server
cd ~/ocr_server
workon django_ocr_server
python manage.py runserver
```

2.3 Centos 7

Install epel repository $sudo yum install epel-release

Install python 3.6

```
$sudo yum install python36
$sudo yum install python36-devel
```

Install gcc

```
$sudo yum intall gcc
$sudo yum install gcc-c++
```

Install dependencies $sudo yum install poppler-cpp-devel

Install tesseract

$sudo yum install tesseract
$sudo yum install tesseract-langpack-rus # install a language pack you need

Install pip $sudo yum install python-pip

Install virtualenv $sudo pip install virtualenv

Create the virtual env for django_ocr_server $sudo virtualenv /var/www/ocr_server/venv -p /usr/bin/python3.6 –distribute

Give rights to the project folder to your user $sudo chown -R {your_user} /var/www/ocr_server/

Activate virtualenv $source /var/www/ocr_server/venv/bin/activate

Install postgresql 11 (The Postgresql version 9.2 that is installing in Centos 7 by default returns an error when applyi

$sudo rpm -Uvh
https://yum.postgresql.org/11/redhat/rhel-7-x86_64/pgdg-redhat-repo-latest.noarch.rpm
$sudo yum install postgresql11-server
$sudo yum install postgresql-devel
$sudo /usr/pgsql-11/bin/postgresql-11-setup initdb
Edit /var/lib/pgsql/11/data/pg_hba.conf
host all all 127.0.0.1/32 md5
host all all ::1/128 md5
$sudo systemctl enable postgresql-11
$sudo systemctl start postgresql-11
$sudo -u postgres psql
create database django_ocr_server encoding utf8;
create user django_ocr_server with password 'django_ocr_server';
alter database django_ocr_server owner to django_ocr_server;
alter user django_ocr_server createdb; # if you want to run tests
q
pip install psycopg2-binary # (on virtualenv django_ocr_server)

Installing django-ocr-server (on virtualenv django_ocr_server). It installs Django as a dependency
$pip install django-ocr-server

Create django project (on virtualenv django_ocr_server)

$cd /var/www/ocr_server
$django-admin startproject ocr_server .

Edit ocr_server/settings.py Add applications to INSTALLED_APPS

Configure database connection

Edit ocr_server/urls.py

Apply migrations (on virtualenv django_ocr_server) $python manage.py migrate

Create superuser (on virtualenv django_ocr_server) $python manage.py createsuperuser

Run server (on virtualenv django_ocr_server), than visit http://localhost:8000/
$python manage.py runserver

CHAPTER 3

Configuration

For changing your django_ocr_server behavior you can use several parameters in the settings.py of your django project.

OCR_STORE_FILES Set it to True (default) to enable storing uploaded files on the server

OCR_FILE_PREVIEW Set it to True (default) to enable showing uploaded images preview in admin interface

OCR_TESSERACT_LANG Sets priority of using languages, default to 'rus+eng'

OCR_STORE_PDF Set it to True (default) to enable storing created searchable PDFs on the server

OCR_FILES_UPLOAD_TO Sets path for uploaded files

OCR_PDF_UPLOAD_TO Sets path for created searchable PDFs

OCR_FILES_TTL Sets time to live for uploaded files, uploaded files older this interval will be removed. Use python datetime.timedelta to set it or 0 (default) to disable.

OCR_PDF_TTL Sets time to live for created searchable PDFs, PDFs older this interval will be removed. Use python datetime.timedelta to set it or 0 (default) to disable.

OCR_TTL Sets time to live for created models of OCRedFile, models older this interval will be removed. Use python datetime.timedelta to set it or 0 (default) to disable.

CHAPTER 4

Deploying to production

4.1 Linux Mint 19 (Ubuntu bionic)

Installing nginx $sudo apt install nginx

Installing uwsgi (on virtualenv django_ocr_server) $pip install uwsgi

Create {path_to_your_project}/uwsgi.ini

Create /etc/nginx/sites-available/django_ocr_server.conf

Enable the django_ocr_server site $sudo ln -s /etc/nginx/sites-available/django_ocr_server.conf /etc/nginx/sites-enabled/

Remove the nginx default site $sudo rm /etc/nginx/sites-enabled/default

Create the systemd service unit /etc/systemd/system/django-ocr-server.service

Reload systemd $sudo systemctl daemon-reload

Start the django-ocr-server service $sudo systemctl start django-ocr-server

Enable the django-ocr-server service to start automatically after server is booted $sudo systemclt enable django-ocr-server

Start nginx $sudo systemctl start nginx

Enable nginx service to start automatically after server is booted $sudo systemctl enable nginx

Go to http://{your_server}:80 You will be redirected to admin page

4.2 Centos 7

Installing nginx $sudo apt install nginx

Installing uwsgi (on virtualenv django_ocr_server) $pip install uwsgi

Create /var/www/ocr_server/uwsgi.ini

Create the systemd service unit /etc/systemd/system/django-ocr-server.service

Reload systemd service $sudo systemctl daemon-reload

Chango user of /var/www/ocr_server to nginx $sudo chown -R nginx:nginx /var/www/ocr_server

Start Django-ocr-server service $sudo systemctl start django-ocr-service

Check that port is up

$sudo netstat -anlpt | grep 8003

you have to got something like this:
tcp 0 0 127.0.0.1:8003 0.0.0.0:* LISTEN 2825/uwsgi

Enable Django-ocr-server uwsgi service $sudo systemctl enable django-ocr-service

Edit /etc/nginx/nginx.conf

Configure selinux

Start nginx service $sudo systemctl start nginx

Enable nginx service $sudo systemctl enable nginx

Configure firewall

$sudo firewall-cmd –zone=public –add-service=http –permanent
$sudo firewall-cmd –reload

Go to http://{your_server}:80 You will be redirected to admin page

CHAPTER 5

Usage examples

You can download all examples from https://github.com/shmakovpn/django_ocr_server/tree/master/usage_examples

5.1 curl

Use curl with '@' before the path of the uploading file

```
#!/usr/bin/env bash
curl -F "file=@example.png" localhost:8000/upload/
```

5.2 python

Use requests.post function

```
import requests


with open("example.png", 'rb') as fp:
    print(requests.post("http://localhost:8000/upload/",
                        files={'file': fp}, ).content)
```

5.3 perl

Use LWP::UserAgent and HTTP::Request::Common

```perl
#!/usr/bin/perl
use strict;
use warnings FATAL => 'all';
use LWP::UserAgent;
use HTTP::Request::Common;

my $ua = LWP::UserAgent->new;
my $url = "http://localhost:8000/upload/";
my $fname = "example.png";

my $req = POST($url,
    Content_Type => 'form-data',
    Content => [
        file => [ $fname ]
    ]);

my $response = $ua->request($req);

if ($response->is_success()) {
    print "OK: ", $response->content;
} else {
    print "Failed: ", $response->as_string;
}
```

5.4 php

Use CURLFile($file, $mime, $name)

```php
<?php
//Initialise the cURL var
$ch = curl_init();

//Get the response from cURL
curl_setopt($ch, CURLOPT_RETURNTRANSFER, 1);

//Set the Url
curl_setopt($ch, CURLOPT_URL, 'http://localhost:8000/upload/');

//Create a POST array with the file in it
$file='example.png';
$mime=getimagesize($file)['mime'];
$name=pathinfo($file)['basename'];
$postData = array(
    'file' => new CURLFile($file, $mime, $name),
);

curl_setopt($ch, CURLOPT_POSTFIELDS, $postData);

// Execute the request
$response = curl_exec(  $ch);
echo($response);

curl_close ($ch);

?>
```

CHAPTER 6

Running tests

Perform under you django_ocr_server virtual environment $python manage.py test django_ocr_server.tests

CHAPTER 7

API documentation

Django-ocr-server provides API documentation use restframework.documentation and swagger. Visit http://localhost:8000/swagger and http://localhost:8000/docs/

CHAPTER 8

Management Commands

Run it to clean trash. It removes all uploaded files and PDFs that do not have related models in database.
$python manage.py clean

Run it to remove models, uploaded files and PDFs, whose time to live (TTL) has expired.
$python manage.py ttl

CHAPTER 9

Creation a distribution package

As mentioned earlier, the automatic installation script ‘install_ubuntu.sh’ uses the package from the PyPI repository by default. To change this behavior or if you need your own distribution package you can build it.

Run command

```
$cd path to cloned project from github
$python setup.py sdist
```

Look in ‘dist’ directory, there is your package was created.

Also you can continue automatic installation. The package will be used.

CHAPTER 10

Indices and tables

- genindex
- modindex

Index