

Tipo : Guía de Laboratorio
Capítulo : Aplicación Inteligente
Duración : 360 minutos (4 horas)

I. OBJETIVO

Desarrollar una aplicación inteligente usando Django.

II. REQUISITOS







Los siguientes elementos de software son necesarios para la realización del laboratorio:

- Instalar Anaconda en Windows
- Navegador web

III. EJECUCIÓN DEL LABORATORIO

- Ejercicio: Desarrollar una aplicación inteligente usando Django.

- 5.1.1 Abrir el command prompt - cmd.exe
- 5.1.2 Verificar instalación de Anaconda
 - a. conda --version
- 5.1.3 Crear entorno virtual
 - a. conda create --name lab2 python=3.5
- 5.1.4 Activar entorno virtual
 - a. activate lab2
- 5.1.5 Instalar django 1.11
 - a. pip install django==1.11
- 5.1.6 Instalar otras librerías
 - a. pip install pandas
 - b. pip install scikit-learn
 - c. pip install nltk
 - d. pip install python-docx
 - e. pip install xlrd
- 5.1.7 Ejecutar django
 - a. Crear la carpeta de trabajo (ej. LABS)
 - b. django-admin startproject lab2
 - c. cd lab2
 - d. python manage.py runserver.

	documents	13/05/2019 07:44 ...	Carpeta de archivos	
	lab2	13/05/2019 07:44 ...	Carpeta de archivos	
	matcher	13/05/2019 07:44 ...	Carpeta de archivos	
	bumeran.xlsx	24/04/2019 06:08 ...	Hoja de cálculo d...	3,623 KB
	db.sqlite3	01/05/2019 04:17 ...	Archivo SQLITE3	39 KB
	manage.py	01/05/2019 03:20 ...	Archivo PY	1 KB

1. Configuración inicial

- Instalar editor atom <https://atom.io/>
- django-admin startapp matcher
- editar lab2/settings.py

```
INSTALLED_APPS = [  
    'matcher',  
    'django.contrib.admin',  
    'django.contrib.auth',  
    'django.contrib.contenttypes',  
    'django.contrib.sessions',  
    'django.contrib.messages',  
    'django.contrib.staticfiles',  
]
```

2. Modelo de datos

- Editar models.py

```
from django.db import models  
  
class Document(models.Model):  
    document = models.FileField(upload_to='documents/')  
    uploaded_at = models.DateTimeField(auto_now_add=True)
```

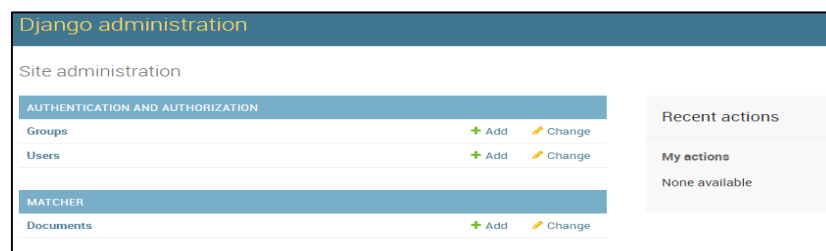
- python manage.py makemigrations
- python manage.py migrate

3. Django Admin

- python manage.py createsuperuser
- Editar admin.py

```
from django.contrib import admin  
from . models import Document  
  
admin.site.register(Document)
```

- 127.0.0.1/admin
- Login



4. Home Page

- a. Editor lab2/urls.py

```
from django.conf.urls import include, url
from django.contrib import admin

urlpatterns = [
    url(r'^admin/', admin.site.urls),
    url(r'^app/', include('matcher.urls')),
]
```

- b. Crear archivo matcher/urls.py

```
from django.conf.urls import include, url

from . import views

urlpatterns = [
    url(r'^$', views.index, name='index'),
]
```

- c. Editor views.py

```
from django.shortcuts import render, redirect
from django.http import HttpResponse

def index(request):
    return render(request, 'index.html')
```

- d. Crear archivo reviews/templates/index.html

```
{% load static %}
<!DOCTYPE html>
<html lang="en" dir="ltr">
  <head>
    <meta charset="utf-8">

    <title>Bienvenido a CVsmart</title>
    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0-beta.2/css/bootstrap.min.css"
    integrity="sha384-PsH8R72JQ350dhVi3uxftmaW6Vc51MKb0q5P2rUppPvrszuE4W1povHYgTpBfshb" crossorigin="anonymous">
  </head>
  <body>
    <div class="jumbotron">
      <h3>Bienvenido </h3>
      

    </div>
  </body>
</html>
```

5. Cargar CV

a. Crear matcher/forms.py

```
from django import forms
from django.forms import.ModelForm
from .models import Document

class DocumentForm(forms.ModelForm):
    class Meta:
        model = Document
        fields = ('document', )
```

b. Editor views.py

```
from .models import Document
from .forms import DocumentForm

def index(request):
    return render(request, 'index.html')

def process_cv(request):
    if request.method == 'POST':
        form = DocumentForm(request.POST, request.FILES)
        if form.is_valid():
            form.save()
            return redirect('index')
    else:
        form = DocumentForm()
    return render(request, 'process_cv.html', {
        'form': form
    })
```

c. Crear matcher/templates/process_cv.html

```
<!DOCTYPE html>
<html lang="en" dir="ltr">
  <head>
    <meta charset="utf-8">
    <title>Upload CV</title>
    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0-beta.2/css/bootstrap.min.css"
    integrity="sha384-PsH8R72JQ350dhVi3uxftmaW6Vc51MKb0q5P2rUpPvrszuE4W1povHYgTpBfshb" crossorigin="anonymous">
  </head>
  <body>
    <h4>Por favor, cargue un CV (docx)</h4>
    <form method="post" enctype="multipart/form-data">
      {% csrf_token %}
      {{ form.as_p }}
      <button type="submit">Upload</button>
    </form>

    <p><a href="{% url 'index' %}">Return to home</a></p>
  </body>
</html>
```

d. Editor index.html

```
<h3>Bienvenido </h3>

<div>
  <p><a href="{% url 'process_cv' %}">PROCESAR CV</a></p>
```

e. Editor matcher/urls.py

```
from django.conf.urls import include, url

from . import views

urlpatterns = [
    url(r'^$', views.index, name='index'),
    url(r'^process/$', views.process_cv, name='process_cv'),
]
```

6. Listar CV

a. Editor views.py

```
import pandas as pd

from . models import Document
from . forms import DocumentForm
```

```
def list_cv(request):
    df = pd.DataFrame(list(Document.objects.all().values()))
    df = df[['id', 'document', 'uploaded_at']]
    df.id = df.id.astype(str)

    def add_url(data):
        output = "<a href='"
        output = output + data
        output = output + "'"
        output = output + data
        output = output + "</a>"
        return output

    df.id = df.id.apply(add_url)

    html_table = df.to_html(
        escape=False,
        index=False,
        border = 1,
        classes = "table table-striped table-hover",
    )
    params = {'html_table': html_table}
    return render(request, 'list_cv.html', params)
```

b. Crear matcher/templates/list_cv.html

```
<!DOCTYPE html>
<html lang="en" dir="ltr">
  <head>
    <meta charset="utf-8">
    <title>LISTA CV</title>
    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0-beta.2/css/bootstrap.min.css"
    integrity="sha384-PsH8R72JQ3S0dhVi3uxftmaW6Vc51MKb0q5P2rRUPvPrszuE4WlpoVHYgTpBfshb" crossorigin="anonymous">
  </head>
  <body>
    <h4>CVs encontrados </h4>
    {{ html_table|safe }}
    <p><a href="{% url 'index' %}">Return to home</a></p>
  </body>
</html>
```

c. Editar index.html

```
<h3>Bienvenido </h3>

<div>
  <p><a href="{% url 'process_cv' %}">PROCESAR CV</a></p>
  <p><a href="{% url 'list_cv' %}">VER CV</a></p>
```

d. Editor matcher/urls.py

```
urlpatterns = [
    url(r'^$', views.index, name='index'),
    url(r'^process/$', views.process_cv, name='process_cv'),
    url(r'^cv/$', views.list_cv, name='list_cv'),
]
```

7. Match empleos

a. Editor views.py

```
def cv_specific(request, cv_id):
    filename = Document.objects.values_list('document', flat=True).get(id=cv_id)
    wordout = read_word(filename)
    jobmat = jobmatrix()
    cvmat = cvmatrix(wordout)
    res = cosine_similarity(cvmatrix, jobmat[0], True)
    res = res[0]
    size = len(res)
    job_simil = pd.DataFrame(columns=('ID', 'Puesto', 'URL', 'Similitud'))
    i = int()
    for i in range(0, size):
        job_simil.loc[i] = [i+1, jobmat[1][i], jobmat[2][i], res[i]]
    sorted_job = job_simil.sort_values(['Similitud'], ascending=False)
    html_match = sorted_job.to_html(
        formatters={
            'Similitud': '{:,.2%}'.format
        },
        index = False,
        col_space = 1000,
        border = 1,
        classes = "table table-striped table-hover",
        escape=False
    )
    params = {'html_match': html_match}
    return render(request, 'match.html', params)
```

```
def read_word(filename):
    doc = docx.Document(filename)
    fullText = []
    for para in doc.paragraphs:
        fullText.append(para.text)
    out = '\n'.join(fullText)
    return out
```

```
def jobmatrix():
    job_raw = pd.read_excel("bumeran.xlsx", encoding='cp1252')
    desc = job_raw.DESCRIPCION
    puesto = job_raw.PUESTO
    url = job_raw.URL
    def desc_to_words(raw):
        letras = re.sub("[^a-zA-ZáóéíúñÑ]", " ", raw)
        words = letras.lower().split()
        stops = set(stopwords.words("spanish"))
        meaningful_words = [w for w in words if not w in stops]
        return( " ".join( meaningful_words ))
    desc_limpio = []
    num_filas = desc.size
    for i in range(0, num_filas):
        desc_limpio.append(desc_to_words(desc[i]))
    desc_limpio = pd.Series(desc_limpio)
    global tfidf_vectorizer
    tfidf_vectorizer = TfidfVectorizer()
    jobmatrix = tfidf_vectorizer.fit_transform(desc_limpio)
    return jobmatrix, puesto, url
```

```
def cvmatrix(wordout):
    letrascv = re.sub("[^a-zA-ZáóéíúñÑ]", " ", wordout)
    minusculascv = letrascv.lower()
    palabascv = minusculascv.split()
    palabascv = [w for w in palabascv if not w in stopwords.words("spanish")]
    resultadocv = " ".join(palabascv)
    desc_limpio_cv = pd.Series(resultadocv)
    cvmatrix = tfidf_vectorizer.transform(desc_limpio_cv)
    return cvmatrix
```

```
import docx
import pandas as pd
import re
from nltk.corpus import stopwords
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity
```


b. Crear matcher/templates/match.html

```
<!DOCTYPE html>
<html lang="en" dir="ltr">
  <head>
    <meta charset="utf-8">
    <title>MATCH CV</title>
    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0-beta.2/css/bootstrap.min.css"
    integrity="sha384-PsH8R72JQ3S0dhVi3uxftmaW6Vc51MKb0q5P2rRUpPvrszuE4W1povHYgTpBfshb" crossorigin="anonymous">
  </head>
  <body>
    <h4>CVs encontrados </h4>
    {{ html_match|safe }}
    <p><a href="{% url 'index' %}">Return to home</a></p>
  </body>
</html>
```

c. Editar matcher/urls.py

```
urlpatterns = [
    url(r'^$', views.index, name='index'),
    url(r'^process/$', views.process_cv, name='process_cv'),
    url(r'^cv/$', views.list_cv, name='list_cv'),
    url(r'^cv/(?P<cv_id>\d+)/$', views.cv_specific, name='cv_specific'),
]
```

8. Pantallazos

Ejecutar en <http://127.0.0.1:8000/app/>



CVs encontrados	
id	document
1	documents/CV_Sergio_Prieto_ES_2019_25mar.docx
2	documents/cv_Cindy_Rosa_Infante_Jaime_10009918...
Return to home	

CVs encontrados			
ID	Puesto	URL	Similitud
6370	INGENIERO DE COSTOS	www.bumeran.com.pe/empleos/ingeniero-de-costos...	21.82%
3650	Líder de proyectos TI -con experiencia en agile	www.bumeran.com.pe/empleos/lider-de-proyectos-...	19.94%
4459	Jefe de Proyecto PMP	www.bumeran.com.pe/empleos/jefe-de-proyecto-pm...	19.60%
3490	Gerente de Proyectos Regionales - San Martín	www.bumeran.com.pe/empleos/gerente-de-proyecto...	18.90%
103	Jefe de Proyectos de Planta	www.bumeran.com.pe/empleos/jefe-de-proyectos...	18.44%

IV. EVALUACIÓN

1. Siendo esto un prototipo ¿qué mejoras se podrían realizar?

a. **Respuesta: Algunas mejoras incluyen:**

- Agregar soporte para cargar CV en PDF.
- Incluir links en la página de match.
- Generar un dashboard a partir de los CVs cargados y match.
- Paginación para los resultados del match.
- Mejorar el tiempo de procesamiento de los resultados que aumenta conforme hay más registros leídos de los empleos
- Incorporar soporte para scraping en tiempo real.
- Poder elegir preferencia de bolsa de empleo.
- Mejoras a nivel de navegación.
- Agregar soporte para responsive app (Smartphone).