**Developer’s Manual**

NLP Recommendation System

**Professor Carlo Lipizzi**

**Kunal Batra**

**Hojat Behrooz**

**Arya Guddemane Vishwakumar**

Table of Contents

[How to start the project - 3 -](#_Toc122372933)

[Directory - 4 -](#_Toc122372934)

[Terminologies - 5 -](#_Toc122372935)

[Tool and Technologies - 6 -](#_Toc122372936)

[User Interface - 6 -](#_Toc122372937)

[Login Page - 6 -](#_Toc122372938)

[Sign Up - 6 -](#_Toc122372939)

[Homepage - 6 -](#_Toc122372940)

[Values and links in each cell - 7 -](#_Toc122372941)

[Results - 8 -](#_Toc122372942)

[Change Password - 9 -](#_Toc122372943)

[Clear Session - 9 -](#_Toc122372944)

[Files on User Page - 9 -](#_Toc122372945)

[Admin’s Page - 9 -](#_Toc122372946)

[Where and how are users created and stored? - 9 -](#_Toc122372947)

[Where and how files and results stored and retrieved? - 10 -](#_Toc122372948)

[Code to upload documents - 12 -](#_Toc122372949)

[Starting point to process document - 13 -](#_Toc122372950)

[System Diagram - 16 -](#_Toc122372951)

# How to start the project

* To start the project, download the project from Github and save it in a new folder.
* Open a terminal at this folder location and install all required libraries listed in requirements.txt

pip install -r requirements.txt

* Open a terminal at this folder and type the command:

python app.py

* This starts the project along with a localhost link displayed on the terminal. Copy and paste that link on to your browser. If this doesn’t work, try changing the port number to available port in your system. The link looks like below

http://localhost:8000/

MongoDB needs to be installed before starting the project. Edit the connection string in the file **app.py** at line number 48 and 52.

To first create a user, login as admin using **admin** as username and **admin** as password.

Create a new user and logout from the admin’s page.

Login using new user credentials.

# Directory

The structure of the project directory is as follows:

main\_folder

|\_\_\_\_\_ Carlo\_ngrams\_tool

| |\_\_\_\_\_\_\_\_\_\_\_\_\_ chunking\_bforce\_plus\_HB\_v2.py

| |\_\_\_\_\_\_\_\_\_\_\_\_\_ chunking\_bforce\_plus\_HB.py

| |\_\_\_\_\_\_\_\_\_\_\_\_\_ chunking\_bforce\_plus\_space\_add.py

| |\_\_\_\_\_\_\_\_\_\_\_\_\_ chunking\_bforce\_plus.py

| |\_\_\_\_\_\_\_\_\_\_\_\_\_ stopwords\_en.txt

|

|\_\_\_\_\_ clustering

| |\_\_\_\_\_\_\_\_\_\_\_\_\_ stores cluster ranks of uploaded documents (.CSV)

|

|\_\_\_\_\_ components

|

|\_\_\_\_\_ static

| |\_\_\_\_\_ css

| | |\_\_\_\_\_\_\_\_\_ index.css

| | |\_\_\_\_\_\_\_\_\_ main.css

| | |\_\_\_\_\_\_\_\_\_ normalize.css

| | |\_\_\_\_\_\_\_\_\_ styles.css

| | |\_\_\_\_\_\_\_\_\_ template.css

|. |\_\_\_\_\_ images

| |\_\_\_\_\_ js

| | |\_\_\_\_\_\_\_\_\_ script.js

| |\_\_\_\_\_ user\_files(files and folders are created when files are uploaded/processed)

|

|\_\_\_\_\_ templates (expanded in next page)

|

|\_\_\_\_\_ app.py

|\_\_\_\_\_ clustering.py

|\_\_\_\_\_ models.py

|\_\_\_\_\_ pdf2text\_tika.py

|\_\_\_\_\_ Recommendation\_system\_v11.py

|\_\_\_\_\_ visualization.py

|\_\_\_\_\_ word2vec\_v1.model

|\_\_\_\_\_ stopwords\_en.txt

Templates folder (expanded)

./templates/

|\_\_\_\_ adminbase.html

|\_\_\_\_ adminlogin.html

|\_\_\_\_ base.html

|\_\_\_\_ changepassword.html

|\_\_\_\_ chpassbyadmin.html

|\_\_\_\_ error.html

|\_\_\_\_ home.html

|\_\_\_\_ login.html

|\_\_\_\_ loginbase.html

|\_\_\_\_ output.html

|\_\_\_\_ showfiles.html

|\_\_\_\_ signup.html

|\_\_\_\_ template.html

|\_\_\_\_ userlist.html

# Terminologies

**Session**: A session is created when either an admin or a user logs in. Session holds information across the website from login to log out. The session is cleared when the user/admin logs out. A session is a dictionary, and the keys are

* **'csrf\_token':** Alpha numeric value
* **'curr\_file':** List of list – contains folder name (string of digits) and name of files that were recently uploaded
* **'files\_saved':** List – contains filenames that are saved while uploading
* **'group':** Dictionary – key: folder name, value: unique id
* **'logged\_in':** Boolen value (True if logged in, else false)
* **'processed':** List – contains file names that are processed
* **'results\_saved':** List – contains names of files who’s results are saved
* **'save\_file':** Boolen value (True if user checks the option “Save File” else False)
* **'save\_result':** Boolen value (True if user checks the option “Save Results” else False)
* **'user':** Dictionary – contains currently logged in information of the user
* **'userfolder'**: string – name of the user folder created at login. All files and folders are created at this location (see project directory)

**Page Tab ID:** This is a counter to track the number of tabs opened for this website. It is initialized to 1 and should always be maintained at 1. When the user copies and pastes the link in another tab, this count increments which results in “restricted access” webpage. This prevents the user from using multiple tabs when the user is logged in. Multiple tabs with different users logged are allowed.

# Tool and Technologies

* Python
* PyMongo, MongoDB
* Flask
* HTML
* CSS
* JavaScript

# User Interface

Template HTML files are present which are being used by other files. These template files are as follows:

* adminbase.html
* base.html
* loginbase.html
* template.html

## Login Page

The username and password are used to check if the user exists. If not, then a message is displayed. Admin can create a user from the admin’s page.

The forms can be found in **login.html** (for user), **adminlogin.html** (for admin). The submitted forms are handled by **script.js** and is searched in the database by flask functions.

## Sign Up

The admin can create a new user from admin’s page. To go to the admin’s page, a link is provided in the login page.

## Homepage

The HTML file associated with homepage is **home.html**

The homepage allows users to upload individual or multiple files. The dropzone allows a user to Drag ‘n’ Drop or browse the files from local storage.

The options “Save Files”, and “Save Results”, are checked by default. The “save\_file” and “save\_result” keys in **session** are assigned to **True** when the user clicks on upload button. If unchecked, it is assigned to **False.** The purpose of this is to determine if the browsed file/s and result/s to be saved to the database.

The **Upload** button is only used to save the file to the database or to the local storage. By default, the files are saved in local storage temporarily until the user signs out.

The **Submit** button takes all the files that were uploaded just before clicking this button. **Session** stores the file information in ‘cur\_file’ and hence can be used to retrieve those files from local storage for processing.

Once the results are out, they are displayed in **Results** section in a form of a table. This is called **DataTables**. Column names of the table are Filename, View Result, Download. A group is created based on how the files were uploaded (individual/multiple). The **group** in session holds this value to create groups based on unique ids created.

### Values and links in each cell

There are multiple scenarios where the user choses to check or uncheck **Save Files** and **Save Results** options before uploading the files. To make it user-friendly, a flexible option to upload files and results are provided from the DataTable.

**Case 1:**

✓ Save Files ✓ Save Results

No action required from DataTable

**Case 2:**

✓ Save Files Save Results

The file is already saved to the database and if the file is processed, a link “upload” is provided to save the result to the database.

**Case 3:**

Save Files ✓ Save Results

The file is not saved but only the result is saved, then this becomes unavailable to the user as the original file is not found in the database.

**Case 4:**

Save Files Save Results

If both these options are unchecked, then a link is provided to upload the file and result to the database. First the file needs to be uploaded and then the result. The link to upload results is disabled unless the original file is uploaded to the database.

**Case 5:**

If the user uploads the documents but does not process it (this happens when the user refreshes the page after uploading the documents, i.e., before clicking “Submit” button), a “Process File” link is provided in the “View Results” column against the name of the file. This takes the file from the local storage and processes it. The results will be available to view, but it requires to be uploaded to the database using the link.

## Results

The files are processed and shown in result section. If multiple files are uploaded, then they are displayed in groups in the datatable. If individual file is uploaded, then that group contains a single file.

**For example:**

Multiple files: File1.pdf, File2.pdf, File3.pdf

These three files are considered as a group and the group number is assigned accordingly.

Individual file: File1.pdf

This file belongs to another group.

Clicking on **view results** in the datatable of a group that has multiple files results in 2 images and an embedded pdf view. The first image is a 3D chart of “Benchmarks clusters similarity to input documents”, the second image is the 2D vertical chart of class similarity of each file. The embedded pdf view is the highlighted document of the original file.

**Clicking on different “View Result” link of the same group results in changing the second image and the embedded pdf view.**

**Clicking on different “View Result” link of different group results in changing all images and the pdf view.**

The class (in second image) can be changed from the dropdown menu by clicking on “Class” button.

These results can be downloaded in the ZIP format. This zip folder contains “Benchmark clusters similarity to input documents”, multiple images of class similarity and multiple highlighted pdf documents (if a group contains multiple files) or one highlighted pdf document (if a group has a single file).

The result in the page “showfiles.html” works the same as it does in “home.html” except that the files are retrieved from the database. (Further explained)

The result section can be cleared by clicking on “Clear Session” or “Sign out” in the homepage. Clicking these will delete all previously uploaded files from the local storage.

## Change Password

Clicking on **Change Password** link opens another page to change the password for the user. The file associated with this is **changepassword.html.** The form is handled by script.js

## Clear Session

Clicking on **Clear Session** link deletes all the values in **session.**

## Files on User Page

The user can view all files previously uploaded in a different page. To navigate into this page, a “Show my files” link is provided in the homepage, just below the dropzone.

The files here are retrieved from the database. Please check the code in “showfile.html” file. Additional options are “Delete File” and “Archive”. Delete file removes that file from the database. Archive makes the file invisible to the user, but can be unarchived by the admin.

## Admin’s Page

The admin has all rights over user. When the admin creates a new user, the details are stored in database and used when the user tries to login. Another link “Show Users” to view all the users is provided. The users are displayed in the datatable with fields such as Name, Email, Files, Password, and Remove User.

The columns Files let the admin view all files of a particular user. The password column lets the admin to change the password of the user and remove user column lets the admin delete the user from the database.

# Where and how are users created and stored?

An user can be created by the admin and the details required are Name, Email and Password.

These details are stored in “users” collection in “user\_login\_system” document in MongoDB. The database name that is assigned in the code can be found in “app.py” file.

Two databases are created, one to store user details, another one to store files and results.

Below image shows the code in “app.py”

Text

Description automatically generated

The database “db” holds user information and “admindb” holds files and results

The collection present in each database is as follows:

“db” = users

“admin” = “fs.files”, “fs.results”

The user information is stored as follows:

Text

Description automatically generated with medium confidence

(screenshot from MongoDB compass)

The information is stored in a JSON format which is easy to retrieve. The user here is “Harry”, email is [harry@gmail.com](mailto:harry@gmail.com), and password is encrypted.

To get information of the user, it is important to use appropriate database i.e., “db”.

# Where and how files and results stored and retrieved?

The files are stored in “admindb” database. The way MongoDB saves is different than other storage systems. The files information is stored in “fs.files” collection and contents of that file is stored in “fs.chunks” collection. “fs.chunks” contains multiple objects that is associated with a single file, whereas “fs.files” contain a single object for a file.

The below screenshot explains it:

Graphical user interface, text, application, Teams

Description automatically generated

(screenshot from “fs.files”)

“\_id” is a unique id created when a file is uploaded to the database. This id is the “files.id” in “fs.chunks”.

“userid” is the unique id of the user that is created when the admin creates a new user. (Refer the user information screenshot)

“group” is a unique id that is created when the user clicks “Upload” button on homepage. This ID is same to all files that are uploaded in a group.

Graphical user interface, text, application

Description automatically generated

(screenshot from “fs.chunks”)

“files\_id” in “fs.chunks” is same as “\_id” in “fs.files”. The above image shows that one file is divided into 3 chunks which can be determined by field “n”. The data is stored in binary format.

# Code to upload documents

The documents are uploaded to local storage and MongoDB simultaneously. The decision to save files to database is determined from the “save\_result” and “save\_files” from **session.** See line 433 in “app.py”

Text

Description automatically generated

The file is uploaded using the following: (see line 440)

mongo.save\_file(f.filename, f, base='fs', userid=curr\_user['\_id'], username=curr\_user['name'], results=session['save\_result'], archived=False, group=str(unique\_grp\_id))

Line 444 onwards: Checks if the folder is created, if not it creates a folder and uploads all documents and results into that folder.

# Starting point to process document

Once the document is uploaded, the user clicks on “Submit” button on homepage. Upon submitting, a function “output\_page()” in the app.py is called.

Text

Description automatically generated

The above function is called from “home.html”. The function “extractAll(user\_file\_path”)” is called from “Recommendation\_system\_v11.py”. The parameter is the path of the original file located in the local storage. “Recommendation\_system\_v11.py” processes the document and saves the highlighted files in the local storage. Furthermore, “visualize()” function is called to create visualization of the results. The data that is used to create visualization is stored in local storage in CSV format. These results are stored in database based on the value of “save\_file” from **session**. Line 491.

The “og\_file\_id” is the original file id that is used to determine the resultant file associated with the original file.

The images that are generated are also stored into the database.

The retrieving of each file uses the “\_id”, “filename”, “username” and “user\_id” .

The retrieving of result files is determined by using “original\_file\_id” which is same as “\_id” of the original file.

Functions such as

* start\_session()
* start\_admin\_session()
* password\_check()
* signup()
* signout()
* login()
* changepassword()
* removeUser()
* deletefile()
* showFiles()
* showFilesbyAdmin()
* adminlogin()
* userlist()
* getallfiles()

are present in **models.py** file, which are all called from **app.py.** The models.py file creates model for each functionality such as getting all files, remove the user, deleting a file etc.

A simple illustration of the follow of job

Database (MongoDB)

Store and retrieve

Website

# System Diagram

Logged in

Admin Page

(Create New User)

Home Page

Recommendation\_system\_v11.py

Local Storage

Database

(MongoDB)

Upload

Submit

Stores by default

If “save files” and/or “save results” is checked

Results

Results

Show Files

User Data

User list

Go to Admin

Retrieves User Data

Retrieves files

Visualzation.py

Results

Login Page