**Custom Chatbot Development**



Session: 2021-2025

**Submitted by:**

Areeba Amjad 2021-SE-17

Kausar Fatima 2021-SE-25

Samiha Shahzad 2021-SE-33

**Submitted to:**

Dr.Farah Adeeba

Department of Computer Science, New Campus

**University of Engineering and Technology**

**Lahore, Pakistan**

**First Deliverable:**

Crawl the data from the URL <https://pkm.punjab.gov.pk/public> and then clean and preprocess the crawl data was our first deliverable. **Introduction:**

Our first deliverable was based on data crawling from the specified website mentioned <https://pkm.punjab.gov.pk/public> and then applying cleaning and preprocessing techniques on that data in order to use the cleaned data for training the chatbot.  
We extract all the data in Urdu and store the extracted data in separate files based on URLs. Each web-page URL has a separate extracted file which stores it’s data in Urdu.

After extraction/crawling of the data we cleaned the data using different techniques. In the cleaning phase we remove the duplication, normalize the data and labeled the data based on the extracted tags.

The libraries or tools we used for crawling and cleaning of the data includes Selenium, chrome-driver, BeautifulSoap. The selenium controls the chrome and run the website dynamically while BeautifulSoap helps it to extract the data.Then for the cleaning we use the library of BeautifulSoap.

**Data Crawling Process:**

The source website to crawl the data is <https://pkm.punjab.gov.pk/public> . This website link takes us to the website where the english content of the website is displaying. As we want to extract only the urdu pages therefore we use the URL <https://pkm.punjab.gov.pk/public/langswitch/switchLanguage/urdu> as the starting url as it takes us directly to the urdu side of that website.  
We created three different files after extracting data using three different codes. The reason for using 3 different codes is due to the different logics for extracting those pages.  
**Tools for the data crawling:**The tools/libraries we used for data crawling includes **BeautifulSoap** and **Selenium.** To use selenium we have installed **chromedriver64.exe**. The selenium runs the website dynamically loads all the pages one by one and then extract all the data in urdu from that page and then store the extracted data in JSON file.

**Separate codes and files:**

Based on the structure of the webpages we have created three different files using three different codes.

* A separate code and file to store extracted data from the url <https://pkm.punjab.gov.pk/public/home/contact_map_urdu>
* A separate code and file to store extracted data for the URL <https://pkm.punjab.gov.pk/public/home/services_urdu?id=a>
* A separate code and file to extract data from the remaining URLs <https://pkm.punjab.gov.pk/public> <https://pkm.punjab.gov.pk/verify/urdu_form> <https://pkm.punjab.gov.pk/track/urdu_form>

**Reason for Separate codes and files:**

* Some webpages contains tabs which have different structure than other webpages.
* Tags require to label the data on the webpages
* Labeled data is stored more correctly in separate files.

**Code Snippets:**

* Set up and import the **selinium webdriver** to browse the website automatically
* import **beautifulSoap** for parsing and extracting data from HTML
* Import **re** to provide regular expression for text cleaning
* Import **json** to store the extracted data in .json format
* Import **time** to allow webpages to load properly before data extraction
* Provide installed chromedriver path “chrome\_driver\_path = r"C:\Users\HP\Downloads\chromedriver-win32\chromedriver-win32\chromedriver.exe""
* Initializing the chromedriver through “service” and “option” objects
* Navigating to the stating url WebDriverWait(driver, 10).until(EC.presence\_of\_element\_located((By.XPATH, "//\*[@id='container']")))
* Identify the XPaths and wait until that xpath found WebDriverWait(driver, 10).until(EC.presence\_of\_element\_located((By.XPATH, "//\*[@id='container']")))
* Navigatng to the specified url to extract data from that page e.g <https://pkm.punjab.gov.pk/public/home/contact_map_urdu>
* Wait until the driver loads tab link according to the specific xpath e.g //ul[@id='myUL']/li/a
* Extract tab links if exist depending upon which page you are currently working on e.g //ul[@id='myUL']/li/a
* Defining function to extract data from each tab using tab index. Automatically clicks the tab wait for 6 sec to load the content and then extract all the data which is visible.

def extract\_tab\_content(tab, tab\_index):

try:

WebDriverWait(driver, 10).until(EC.element\_to\_be\_clickable(tab))

driver.execute\_script("arguments[0].scrollIntoView(true);", tab)

driver.execute\_script("arguments[0].click();", tab)

* Parse and extract the data using beauitiful soap

page\_content = driver.page\_source

soup = BeautifulSoup(page\_content, "html.parser")

* Write a loop to traverse through all the tabs and store extracted data in JSON file.

**Data Cleaning Process:**

**Tools for the data crawling:**We parsed the html content by using **BeautifulSoap** to make the extraction and traversal easy so that it will only extract the structured data.We use **re** to retain only valid characters and remove the character we don’t want.

**Code Snippets:**

* Use beautifulSoap to extract the data by parsing the HTML content e.g soup = BeautifulSoup(page\_content, "html.parser")
* Clean the text by removing the special characters or such characters we don’t want. We have remove all the special characters except @, . -, and numbers so that we can include email and phone numbers. All the remaining characters will be removed. We will also remove the extra spaces from the text.

def preprocess\_text(text):

text = " ".join(text.split())

text = re.sub(r'[^\u0600-\u06FFa-zA-Z0-9\s.@]', '', text) except Urdu, English, numbers, ".", and "@".

return text

* The redundant or duplicate data is removed by and the irrelevant data is ignored

if text:

if text not in tab\_data[last\_heading]:

tab\_data[last\_heading].append(text)

* There exist many heading that are repeating in the extracted data multiple times. We remove those heading and their content by first identifying them through tags and then manually including them in the code to extract the heading or it’s content wherever it occurs.

Excluded\_headings = ["خدمت مراکز", "تعارف", "ہماری سروسز", "ہم سے رابطہ کریں", "Contact Details", "اہم خصوصیات"]

if last\_heading in excluded\_headings:

last\_heading = None

* Label the data for easy retrieval for later use.

**For li tags**

elif last\_heading and element.name == 'li':

text = preprocess\_text(element.get\_text(strip=True))

if text and text not in tab\_data[last\_heading]:

tab\_data[last\_heading].append(text)

**For other tags**

elif last\_heading and element.name in ['p', 'ul', 'ol', 'div']:

text = preprocess\_text(element.get\_text(strip=True))

if text and text not in tab\_data[last\_heading]:

tab\_data[last\_heading].append(text)