Acadustia
(Data Analytics in Education)

Final Year Project Report
By

Shah Mashud Alam (00000142522)
Hafiz Adnan Iqbal Khan (00000121846)
Muhammad Ehsan ul Haq (00000125034)

In Partial Fulfillment
Of the Requirements for the degree
Bachelors of Engineering in Software Engineering (BESE)

School of Electrical Engineering and Computer Science National University of Sciences and Technology Islamabad, Pakistan
(2019)

DECLARATION

We hereby declare that this project report entitled “Acadustia (Data Analytics in Education)” submitted to the “SEECS”, is a record of an original work done by us under the guidance of Supervisor “Dr. Muhammad Ali Tahir” and that no part has been plagiarized without citations. Also, this project work is submitted in the partial fulfillment of the requirements for the degree of Bachelor of Computer Science.

Team Members	**Signature**
Shah Mashud Alam	_____
Hafiz Adnan Iqbal Khan	_____
Muhammad Ehsan ul Haq	_____

Supervisor:	**Signature**
Dr. Muhammad Ali Tahir	_____

Date: _____
Place: _____

Mr. Taufeeq Ur Rehman	_____

Date: _____
Place: _____

DEDICATION

This thesis is dedicated to my father, who taught me that the best kind of knowledge to have is that which is learned for its own sake. It is also dedicated to my mother, who taught me that even the largest task can be accomplished if it is done one step at a time.

ACKNOWLEDGMENTS

I would like to express my special thanks of gratitude to my teachers who gave me the golden opportunity to do this wonderful project on the topic Data Analytics in Education, which also helped me in doing a lot of Research and I came to know about so many new things I am really thankful to them.

Secondly, I would also like to thank my parents and friends who helped me a lot in finalizing this project within the limited time frame.

TABLE OF CONTENT

ABSTRACT

One of the biggest issues a university graduate faces is his professional career is unemployment or unpaid internship, but when we go deeper in this issue, we find that the reason for it is that what industry is demanding is usually not present the fresh graduate and this is because of the gap between academia and industry. In our project, we will try to solve this problem through data mining and data analytics.

Chapter 1

INTRODUCTION

Unemployment is one of the biggest issues that Pakistan is facing. Moreover, unemployment of skilled labor is another heartbreaking thing as more than five hundred thousand university graduates are unemployed. So, when we look deeper inside, we get to know the reason behind this is the unmatched demand and supply between academia and industry, as what industry demands is sometimes not present in the students or they lack in that field. Secondly, another issue we face in our society is lacking in foreseeing the future trends of industry and technologies. Lastly, universities and educational institutes want to know about the specializations they have and secondly, they also want to know the areas in which they need to promote specializations.

In our project, we will be connecting industry with academia by extracting the job data posted on online portals after this we will be cleaning data to remove the redundancies and repetitions. After this we will be analyzing the most demand skill set required by the industry, secondly we will be predicting the future trends which the industry might take up and lastly we will be analyzing university profiles by their given data through which we can help in suggesting the fields in which the universities can promote research and development in order to improve their technological profile.

Chapter 2

LITERATURE REVIEW

A lot of work is been done in the field of data analytics in various fields like in marketing, stocks and etc., but very few works are being done in the field of education technology (EdTech). One of the reasons for this lacking is that data analytics in education is not very much profitable in the monetary term but it can create a huge impact on the world through effective learning and better education.

Few works are done in this field but they lack in the application in real-world scenario. Moreover

- IBM Analytics

  IBM has its own project that has been using analytics and helping schools succeed. The University of Florida has also been using this platform to extract the student data and use it to monitor and predict their student performance. Finger Lakes Community College and The Keller Graduate School of Management have already adopted the IBM Analytics platform to track their students and both the institutes have seen a rise in the performance of their students.

- Adaptive Learning

  Arizona State University's (ASU) math's department has adopted a system called 'adaptive learning' to improve their student performance. Their systems collect information of students like marks, strengths, weaknesses and even measures moments of hesitation. ASU's eAdvisor integrates learning management system information about student activity with registration data and student background characteristics and they are notified when a student gets off track so that they can help them before anything major happens. Adopting this system, ASU has improved its student success rate and its dropout rate has decreased to about 54%. The University of Nevada also collects student learning and navigational patterns to analyze past and present data and enhance the student experience.

- Georgia State University (GSU)

  Georgia State University (GSU) mining the student data to identify courses in which they showed poor performance and created a supplemental instruction program to help with those courses. By doing this, they found that its graduation rate went up from 32% to 54% between the year 2003

and 2014, on adopting data to solve issues of retention and course completion.

- University of Nevada's System

  Collects data on student learning to improve lecture deliverance

For prediction, we use Recurrent Neural Networks, Long Short-Term Memory in particular. Time Series Forecasting is a widely used technique in which we try to predict n + 1 value from a given data. Here we use it to predict the future trend.

Named Entity Recognition is also another widely used technique in Natural Language Processing, here we use it to extract data from websites.

Chapter 3

PROBLEM DEFINITION

The gap between industry demand and academia supply. And shortcomings in comprehending the future trends.

Companies tend to avoid untrained employees as they will have to invest in them in the form of training sessions and salary. That's why they opted for experienced people.

So basically, unmatched demand of industry and supply of academia is a great issue, and the reason behind this is the gap between academia and industry, as those skills which are being demanded by the industry is sometimes not taught in universities or students are also not aware them. Secondly, it is observed that trends are first formed in west and then they are followed in rest of the world, so we will be predicting the trends which can come to Pakistan and the students can benefit from it. Thirdly, there are disciplines in universities which need more research and development so we will be analyzing data in order to find out those fields which need more efforts.

Chapter 4

METHODOLOGY

In order to analyze the data, we first need to collect from some source. There could be various options to choose from like publicly available datasets, but for this domain, there is no dataset available. We require two types of data, first is regarding faculty specialization, and the other is regarding job trends. Both of the types of data need some reliable source.

We decided to scrape data from the internet. As for the sites from which we will scrape data, we selected official university sites for faculty related data and for industrial jobs data we selected job posting sites.

In websites there are two main categories, one is regular websites that do not rely on JavaScript heavily and one can easily obtain data from them just by using any regular HTML or XML parser. The other type of websites is those that use modern JavaScript frameworks such as React, Angular, etc. These cannot be parsed regularly, for this we used the concept of headless browser or browser automation in which we mimic user clicks on different components and extract data from them.

All the data extraction techniques that are discussed above are commonly used but have a big disadvantage that for large data collection, data from one website is not enough and one parser cannot work for different websites. For this, we introduce a new concept of Automatic Data Parsing. In this, we create a generalized parser to collect all the visible text of a site and feed it to a neural network to classify each term as a Named Entity. The above concept is called Named Entity Recognition.

Once the data is extracted successfully, we visualized the data using a number of charts, for this the library we used was google charts, which is one of the states of the art data visualization library. For the front end, reactive programming was used which is an upgraded version of Observer pattern previously used in creating Graphical User Interfaces.

For prediction, we used Google Trends data, which consists of data from the past 15 years. We could have used a simple technique to predict future trends like logistic regression, but we used state of the art deep learning algorithm called Long Short-Term Memory.

Chapter 5

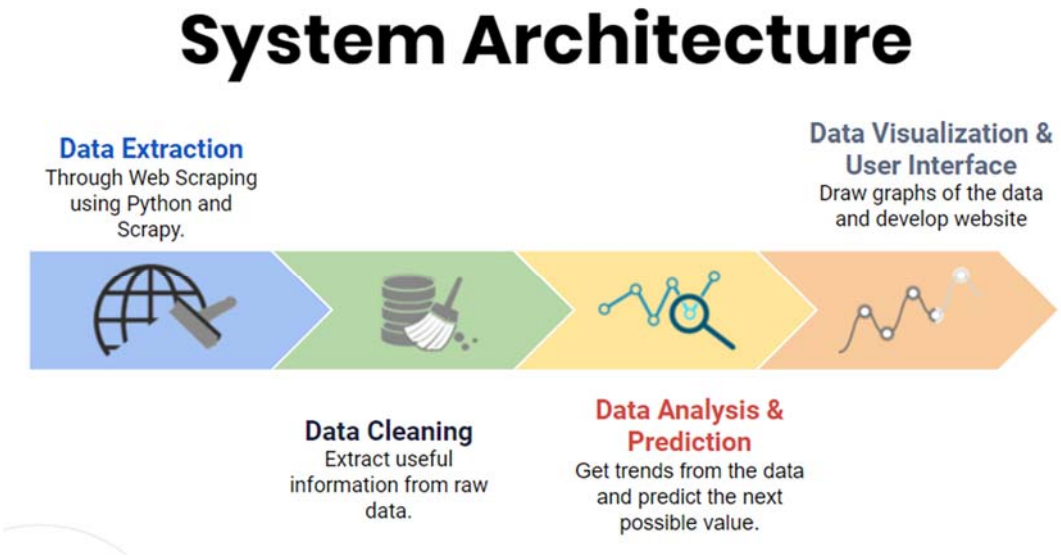
DETAILED DESIGN AND ARCHITECTURE

5.1 SYSTEM ARCHITECTURE

5.1.1 Architecture Design Approach

We followed bottom to top design approach in which we first designed smaller components which will be used to collect data and information, then after this, we moved forward in which we when to toward data purification and cleaning stage in which we cleaned the data to form trends and evaluations. After this, we moved forward and started implementing the analytics part

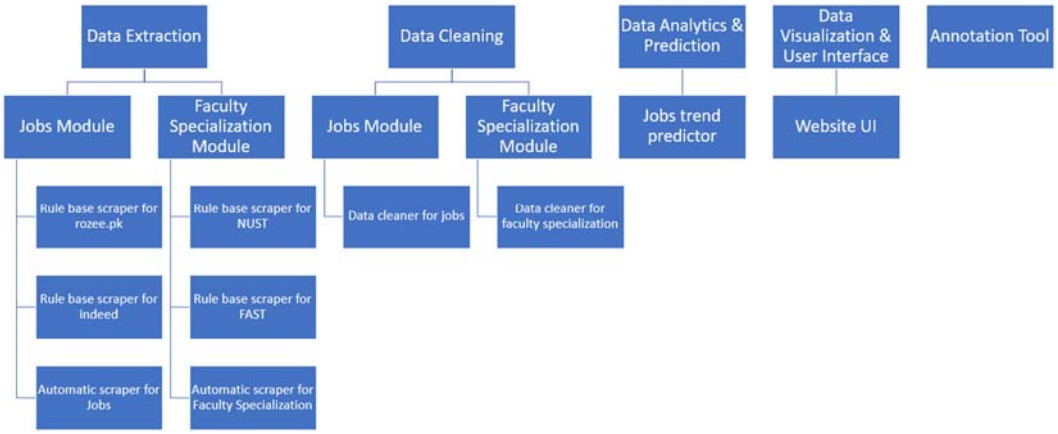
5.1.2 Architecture Design



The System Architecture is divided into four basic parts Data extraction, Data Cleaning, Data analytics, and Data Visualization. In the Data Extraction module, the core responsibility in this module is to collect data from online job portals, like rozee.pk, indeed.us, and LinkedIn. This module is very important as our whole project revolves around data analytics so data is a major requirement. After the data is collected then it is passed to cleaning phase in which Data cleaning module comes in to play, in this module we remove inconsistencies and redundancies from the data, making it readable and understandable. When the data available in a presentable form then it is used to form prediction and analysis regarding trends and requirements. In the last, the processed information is

presented in the form of graphs and illustrations which are easily understandable to people.

5.1.3 Subsystem Architecture



5.2 DETAILED SYSTEM DESIGN

5.2.1 Rule-Based Scraper for Rozee

Classification

Module

Definition

Rule bases scraper to extract jobs data from rozee.pk.

Responsibilities

- Crawling web pages
- Extracting information from the web page using XPath selector

Constraints

For each information required from the webpage, we should be hard code its position on the webpage. Whenever the website administrator changes to web page structure we should change over the code.

Composition

- Crawler
    - Use to traverse web pages.
- Xpath Selector
    - Use to extract the information from a web page

Uses/Interactions

The output of this component is used by “Data cleaner for jobs”.

Resources

- ➔ Libraries
  - ◆ Scrapy
  - ◆ Selenium
  - ◆ XPath Selector
- ➔ Other Resources
  - ◆ Active internet connection

Processing

Go to the base URL provided as input and then crawl all URLs (which may have jobs related info) and extract the content of that page. For extraction, we build two versions of this component.

First one simply gets HTML of the page by HTTP GET request (use scraps for this). But later the website (rozee.pk) converted its view component to the client side. Which means on HTTP GET request we only get a JavaScript code which can render actual content in the browser. To overcome this issue, we used selenium to run that scraped JavaScript code in a headless Chrome browser. Which render the desired HTML content.

After getting the content we parse it using the XPath Selector to extract the information about the job (title, company name, city, Country, Date of posting, Experience required, Salary, Description)

Time Complexity: O(n*m)
Space Complexity: O(n*m* k)

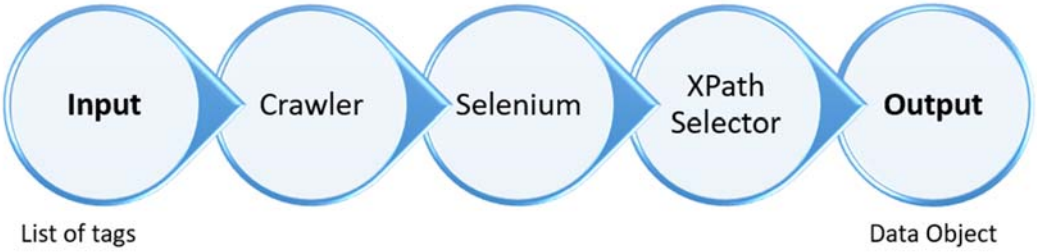
where:

- n: numbers of keywords provided as input
- m: numbers of jobs available for a keyword
- k: number of features extracted (we use k=8)

Interface/Exports

INPUT: a list of tags (keywords) of which the jobs information is required
OUTPUT: a JSON object contains keys (keyword provided) and value (information extracted)

Detailed Subsystem Design



5.2.2 Rule base scraper for indeed

Classification

Module

Definition

Rule bases scraper to extract jobs data from indeed.com.pk.

Responsibilities

- Crawling web pages
- Extracting information from web page

Constraints

Limited features are provided by the library used.

Composition

- Crawler
  - Use to traverse web pages.
- Extractor
  - Use to extract the information from a web page
- Linker (muterun_js)
  - Use to run node jess library from python code

Uses/Interactions

The output of this component is used by “Data cleaner for jobs”.

Resources

➔ Libraries
- Indeed-scraper
- muterun_js

- ➔ Other Resources
  - ◆ Active internet connection

Processing

Python code loop through the provided list of tags (keywords). For each tag its call the node js script through "muterun_js" and pass the tag as a command line argument. Node js script uses that tag as a query in "indeed-scraper" queryOptions. "indeed-scraper" return a JSON object containing the information about available jobs related to that tag. We logged that info to console, from where the python script captures the JSON object and store this data in a dictionary with the tag as the key.

Time Complexity: O(n*m)
Space Complexity: O(n*m* k)

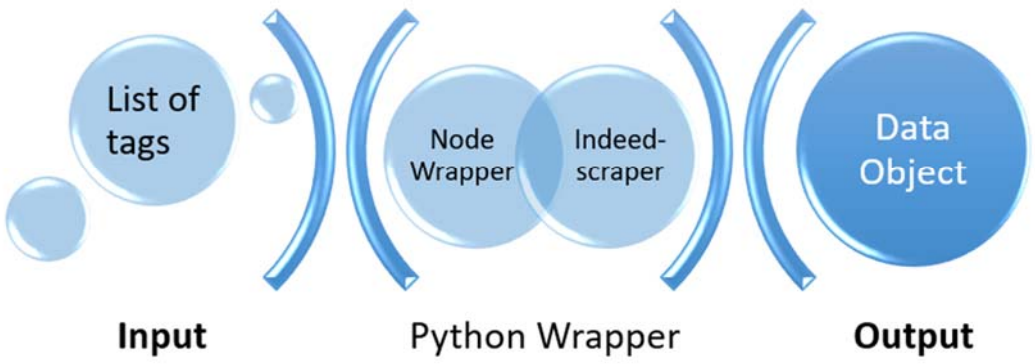
where:

- n: numbers of keywords provided as input
- m: numbers of jobs available for a keyword
- k: number of features extracted (we use k=7)

Interface/Exports

INPUT: a list of tags (keywords) of which the jobs information is required
OUTPUT: a JSON object contains keys (keyword provided) and value (information extracted)

Detailed Subsystem Design



5.2.3 Automatic Scraper for Jobs

Classification

Module

Definition

Automatic scraper for any online job portal using machine learning.

Responsibilities

- Crawling web pages
- Extracting visible text
- Apply NER tagger on visible text to extract required data

Constraints

This uses NER (Named Entity Recognition) tagger, a natural language processing technique. Which uses machine learning to propose a tag for each word in the given text. As this is a machine learning technique it's may propose an incorrect tag. So, the changes in error were always there.

Composition

- Crawler
  - Use to traverse web pages.
- Visible Text Extractor
  - Use to extract plan visible text from the webpage
- NER tagger
  - Use to propose a tag for each word in the given text
- Post processor
  - Use to combine the words for each tag and output them as a JSON object

Uses/Interactions

The output of this component is used by "Data cleaner for jobs".

Resources

➔ Libraries
  - Scrapy
  - Selenium
  - NLTK

- ◆ Stanford Named Entity Recognizer

➔ Other Resources

- ◆ Active internet connection

Processing

Python script loops through the list of tags that are given as input. For each tag, crawler downloads the webpages and open them through selenium so the HTML content can capture. Now the job of visible text extractor, it removes the irrelevant data from the HTML content (e.g. HTML tags, comments, JS code, CSS styles, etc.).

Then pass that extracted visible text to NER tagger so it can predict the most relevant tag for each word. These results will feed to the post processor so it can transform them into a presentable form (JSON object).

Time Complexity: O(n*m*p*k)
Space Complexity: O(n*m*p*q*k)

where:

- n: numbers of keywords provided as input
- m: numbers of jobs available for a keyword
- p: numbers of words in the visible text
- q: number of activation parameters
- k: number of features extracted (we use k=2)

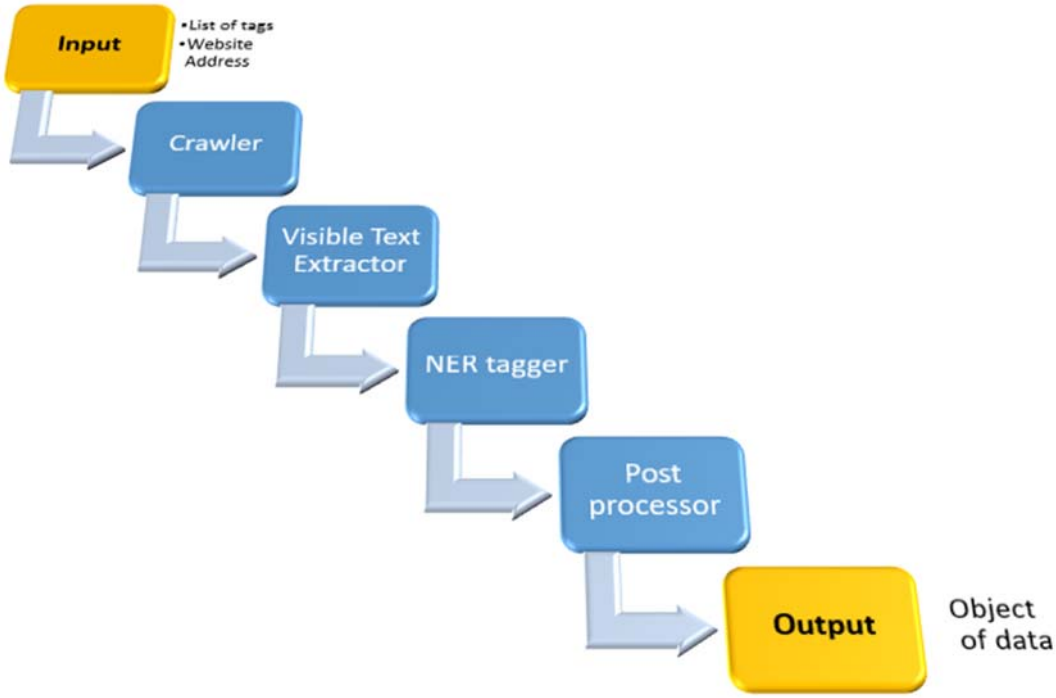
Interface/Exports

INPUT:

- a list of tags (keywords) of which the jobs information is required
- Website address (e.g. rozee.pk or indeed.com.pk)

OUTPUT: a JSON object contains keys (keyword provided) and value (information extracted)

Detailed Subsystem Design



5.2.4 Annotation Tool

Classification

Sub System

Definition

A general-purpose web-based graphical user interface for text annotation. For demo visit https://annotationtool.ml/

Responsibilities

- Track the user click
- Allow user to add new Text
- Allow user to add new Tag
- Show / Download output as a JSON object

Constraints

This uses NER (Named Entity Recognition) tagger, a natural language processing technique. Which uses machine learning to propose a tag for each word in the given text. As this is a machine learning technique it's may propose an incorrect tag. So, the changes in error were always there.

Composition

- Main Panel
    - Use to show the original text and allow the user to click any word to add to the selected tag's list.
- Tag Panel
    - Use to select a tag.

Uses/Interactions

The output of this component is used by "NER Tagger" for training the model.

Resources

➔ Libraries
- jQuery
- Bootstrap

Processing

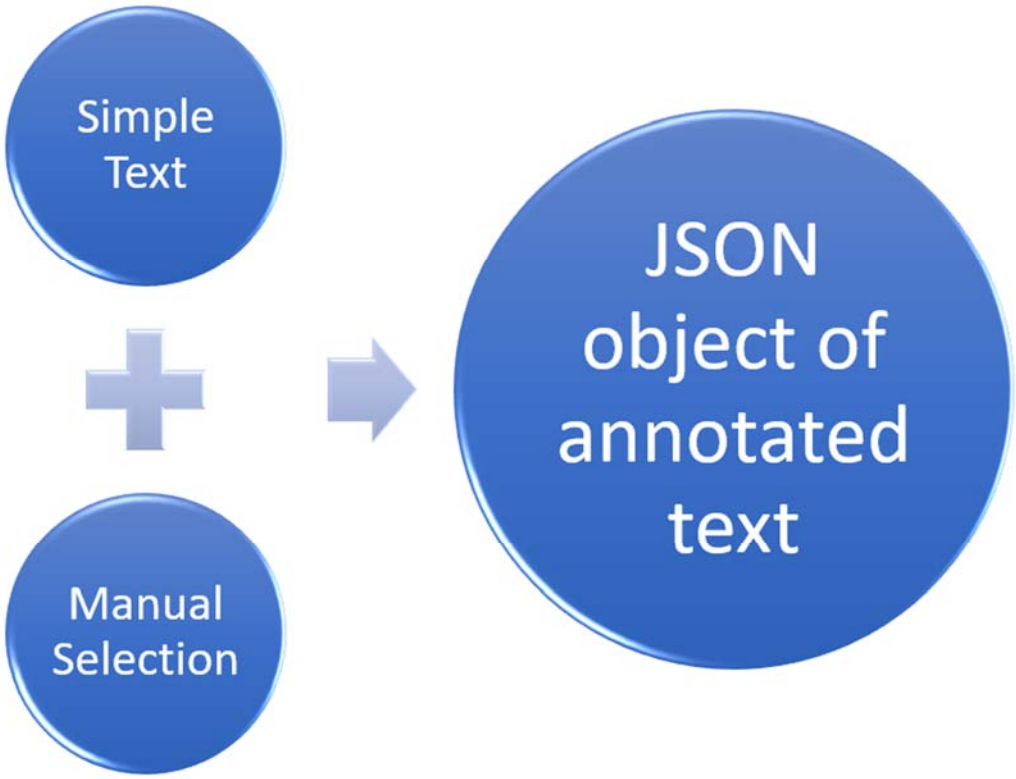
Firstly, the user adds a text which is required to be annotated. Then add desired tags to the tag list. Select a tag, the selected tag is shown above the main panel. Now simply click all the words that are related to the selected tag. To unselect a word right clicks on the same word.

When a user is complete the annotation, he/she simply click "show JSON" button to show or "Save JSON" to download JSON as a file.

Interface/Exports

INPUT: Plain text which is required to be annotated
OUTPUT: a JSON object contains keys (Tags) and value (list of annotated words)

Detailed Subsystem Design



5.2.5 Rule-Based Scraper for NUST

Classification

Module

Definition

Rule bases scraper to extract faculty specialization data from NUST website.

Responsibilities

- Crawling web pages
- Extracting information from the web page using XPath selector

Constraints

For each information required from the webpage, we should be hard code its position on the webpage. Whenever the website administrator changes to web page structure we should change over the code.

Composition

- Crawler
    - Use to traverse web pages.

- Xpath Selector
    - Use to extract the information from a web page

Uses/Interactions

The output of this component is used by "Data cleaner for faculty specialization".

Resources

- ➔ Libraries
    - ◆ Scrapy
    - ◆ XPath Selector
- ➔ Other Resources
    - ◆ Active internet connection

Processing

Go to the base URL provided as input and then crawl all URLs (which may have faculty specialization related info) and extract the content of that page using the XPath Selector library. Output the extracted data as a JSON object containing a list of faculty objects with value have an object contains (name, specialization, department, education, designation)

Time Complexity: O(n)
Space Complexity: O(n*m)

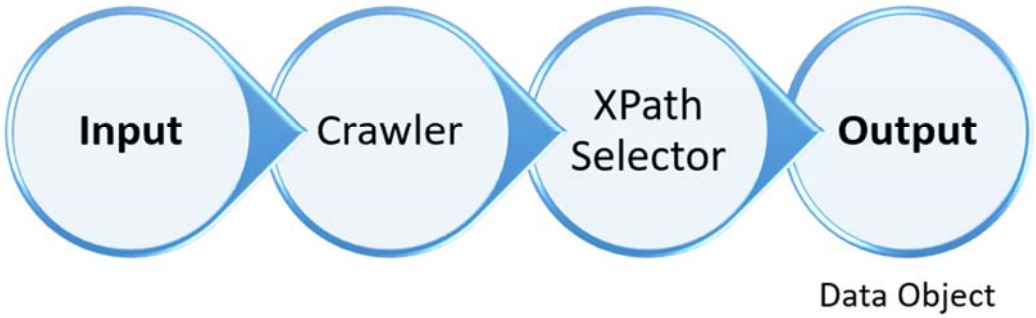
where:

- n: numbers of faculty members in the university
- m: number of features extracted (we use k=5)

Interface/Exports

INPUT: None
OUTPUT: a JSON object contains a list of objects. One object for each faculty member.

Detailed Subsystem Design



5.2.6 Rule-Based Scraper for FAST

Classification

Module

Definition

Rule bases scraper to extract faculty specialization data from FAST website.

Responsibilities

- Crawling web pages
- Extracting information from the web page using XPath selector

Constraints

For each information required from the webpage, we should be hard code its position on the webpage. Whenever the website administrator changes to web page structure we should change over the code.

Composition

- Crawler
  - Use to traverse web pages.
- Xpath Selector
  - Use to extract the information from a web page

Uses/Interactions

The output of this component is used by "Data cleaner for faculty specialization".

Resources

➔ Libraries

- Scrapy
- XPath Selector

➔ Other Resources

- Active internet connection

Processing

Go to the base URL provided as input and then crawl all URLs (which may have faculty specialization related info) and extract the content of that page using the XPath Selector library. Output the extracted data as a JSON object containing a list of faculty objects with value have an object contains (name, specialization, department, education, designation)

Time Complexity: O(n)
Space Complexity: O(n*m)

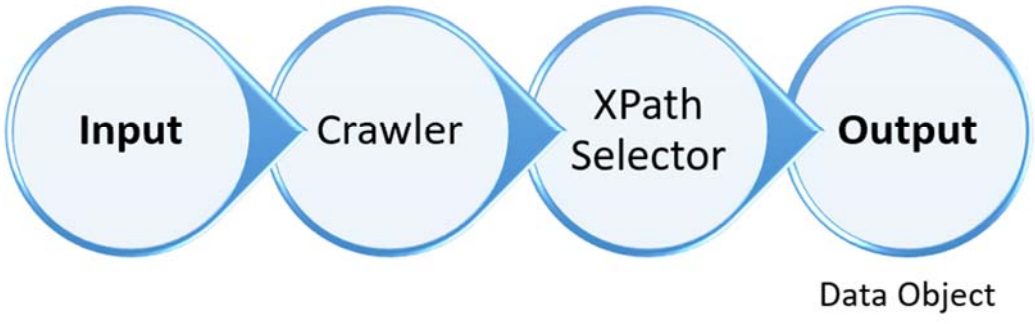
where:

- n: numbers of faculty members in the university
- m: number of features extracted (we use k=5)

Interface/Exports

INPUT: None
OUTPUT: a JSON object contains a list of objects. One object for each faculty member.

Detailed Subsystem Design



5.2.7 Automatic scraper for Faculty Specialization

Classification

Chapter 5

Module

Definition

Automatic scraper for any online job portal using machine learning.

Responsibilities

- Crawling web pages
- Extracting visible text
- Apply NER tagger on visible text to extract required data

Constraints

This uses NER (Named Entity Recognition) tagger, a natural language processing technique. Which uses machine learning to propose a tag for each word in the given text. As this is a machine learning technique it's may propose an incorrect tag. So, the changes in error were always there.

Composition

- Crawler
  - Use to traverse web pages.
- Visible Text Extractor
  - Use to extract plan visible text from the webpage
- NER tagger
  - Use to propose a tag for each word in the given text
- Post processor
  - Use to combine the words for each tag and output them as a JSON object

Uses/Interactions

The output of this component is used by "Data cleaner for jobs".

Resources

- ➔ Libraries
  - ◆ Scrapy
  - ◆ Selenium
  - ◆ NLTK
  - ◆ Stanford Named Entity Recognizer
- ➔ Other Resources
  - ◆ Active internet connection

Processing

Firstly, script crawl towards the faculty pages of the given website. Then download the webpages and open them through selenium so the HTML content can capture. Now the job of visible text extractor, it removes the irrelevant data from the HTML content (e.g. HTML tags, comments, JS code, CSS styles, etc.).

Then pass that extracted visible text to NER tagger so it can predict the most relevant tag for each word (currently model is trained for only faculty member's name and specialization). These results will feed to the post processor so it can transform them into a presentable form (JSON object).

Time Complexity: O(n*m*k)
Space Complexity: O(n*m*p*k)

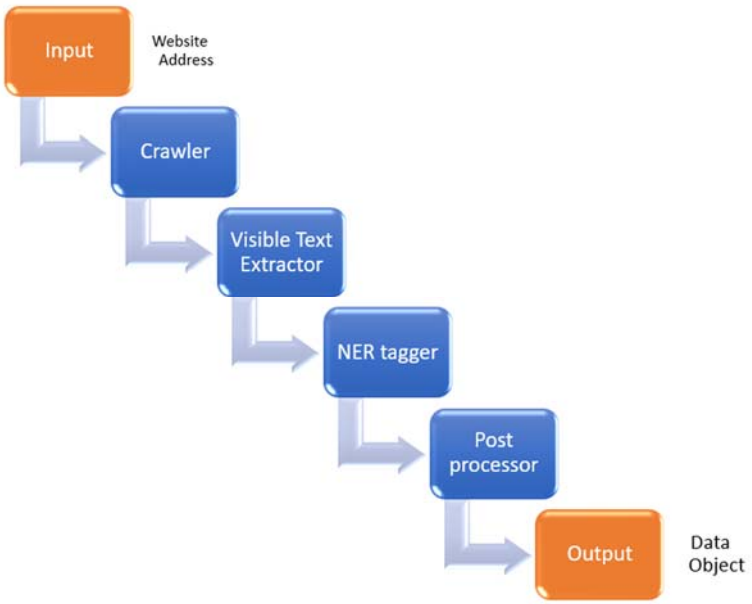
where:

- n: numbers of faculty members in the university
- m: numbers of words in the visible text
- p: number of activation parameters
- k: number of features extracted (we use k=2)

Interface/Exports

INPUT: Website address (e.g. nust.edu.pk or nu.edu.pk/)
OUTPUT: a JSON object contains a list of objects. One object for each faculty member.

Detailed Subsystem Design



5.2.8 Data cleaner for jobs

Classification

Module

Definition

Clean the jobs data acquired from Rule-based scraping or automatic scraping module.

Responsibilities

- Remove incomplete data
- Remove redundant data

Constraints

No constraints.

Composition

- Redundant data remover
  - Use to remove redundant data
- Incomplete data remover
  - Use to remove incomplete data

Uses/Interactions

The output of this component is used by “jobs trend predictor”.

Resources

➔ Libraries
  - ◆ Pandas

Processing

Firstly, convert the dictionary to pandas DataFrame and then remove the incomplete rows then take unique rows to convert them back to JSON and output the result

Time Complexity: O(n*m*k)
Space Complexity: O(n*m*k)

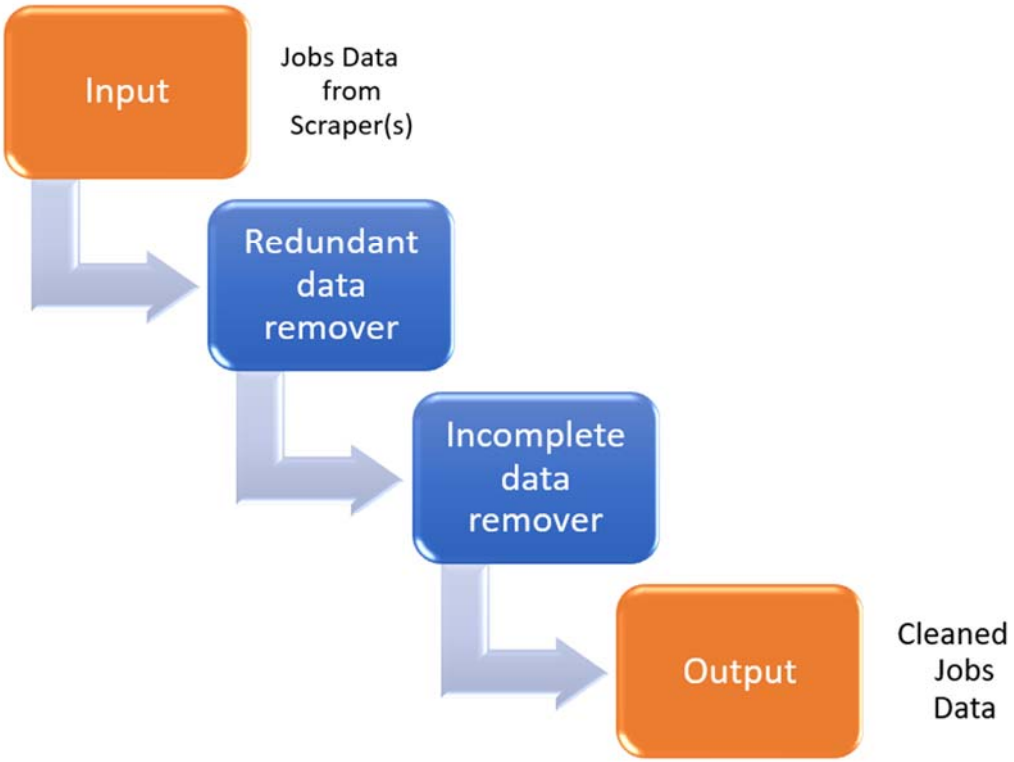
where:

- n: numbers of tags
- m: numbers of jobs
- k: number of features in each job object

Interface/Exports

INPUT: jobs data from Scraper(s) in the form of the JSON object
OUTPUT: a JSON object contains cleaned jobs data

Detailed Subsystem Design



5.2.9 Data cleaner for faculty specializations

Classification

Module

Definition

Clean the faculty specializations data acquired from Rule-based scraping or automatic scraping module.

Responsibilities

- Remove incomplete data
- Remove redundant data

Constraints

No constraints.

Composition

- Redundant data remover
  - Use to remove redundant data
- Incomplete data remover
  - Use to remove incomplete data

Uses/Interactions

The output of this component is used by “Website UI”.

Resources

➔ Libraries
  - ◆ Pandas

Processing

Firstly, convert the dictionary to pandas DataFrame and then remove the incomplete rows then take unique rows to convert them back to JSON and output the result

Time Complexity: O(n*m*k)
Space Complexity: O(n*m*k)

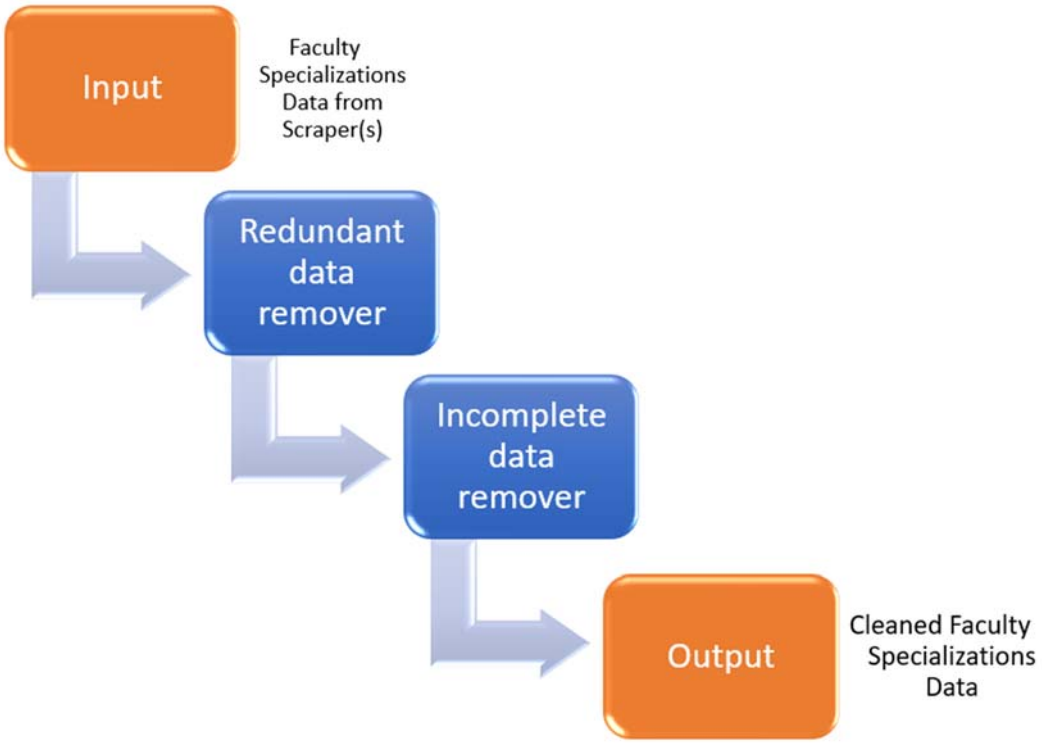
where:

- n: numbers of tags
- m: numbers of jobs
- k: number of features in each faculty member’s object

Interface/Exports

INPUT: faculty specializations data from Scraper(s) in the form of the JSON object
OUTPUT: a JSON object contains cleaned faculty specializations data

Detailed Subsystem Design



5.2.10 Jobs Trend Predictor

Classification

Module

Definition

Predicting the future of a domain using machine learning.

Responsibilities

- Train a separate model for each tag (domain)
- Future prediction using the trained models

Constraints

As this uses machine learning for the prediction. There is always a chance of error in the prediction.

We used Neural networks (LSTM) which is computationally expensive.

Composition

- Trainer
  - Use to train a separate model for each domain.
- Predictor
  - Use to future prediction.

- Post processor
    - Use to combine the predicted values and output them as a JSON object

Uses/Interactions

The output of this component is used by "Website UI".

Resources

➔ Libraries
- Keras
- Tensorflow

Processing

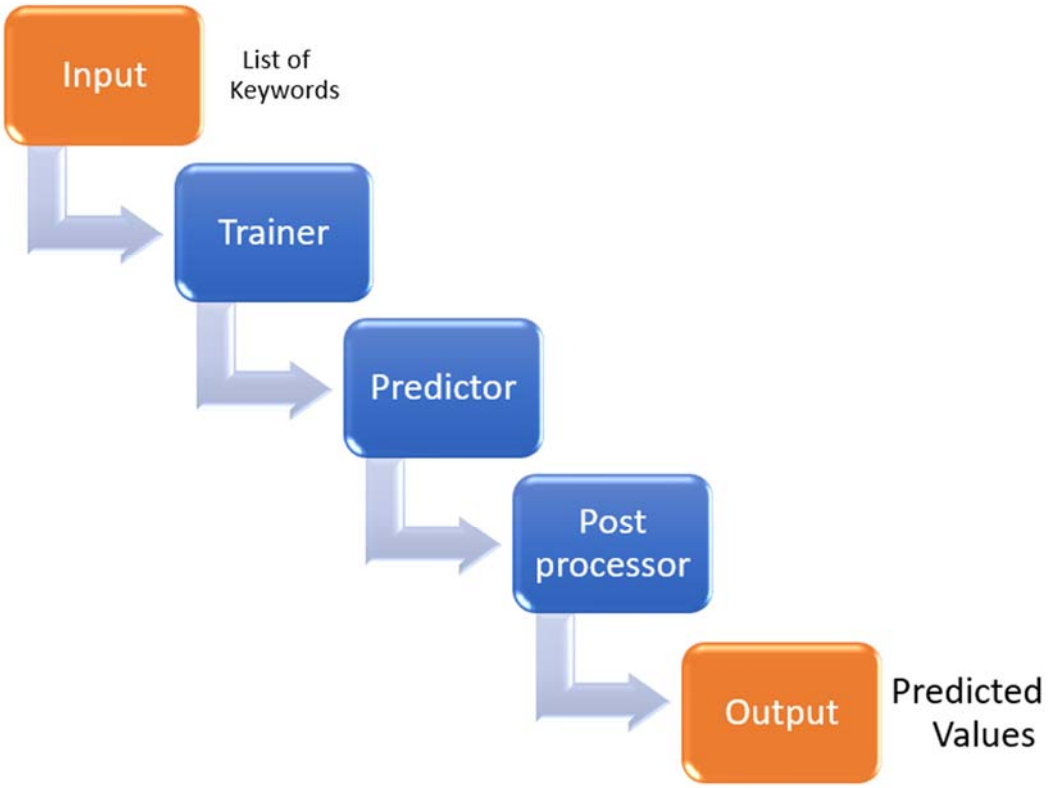
, Ideally, we should train models on data extracted by the online job portals. But these websites do not maintain the record of previous data. They only provide information about jobs available today. And to predict the trend in the next 4 years we should have more the 4 years of data for training. To get this amount of data from these job portals we should run the scraper for 4 years minimum which is not possible. So, we get 14 years of data from google trends. This is not the data of jobs available but the frequency of search of that keyword. As the search frequency also reflect the trends in the market. So, we use it to train the models.

When the models are trained. Predictor used it to predict trends for the next 4 years by an interval of 6 months. Then post processor reform the predicted values along with the respective mean squared error into a JSON object.

Interface/Exports

INPUT: a list of tags (keywords) of which the jobs information is required
OUTPUT: a JSON object contains predicted values for given tags

Detailed Subsystem Design



5.2.11 Website User Interface

Classification

Subsystem

Definition

The presentation layer of the project. It shows the data collected and predicted in an understandable manner.

For demo visit https://acadustia.ml/

Responsibilities

- Search tags to view its data
- Show the data in a visual form which can easy to understand by anyone.

Constraints

No constraints. As this is a website so anyone having an active internet connection can view it.

Composition

- Database

- Use to store data generated from different modules.

- UI
  - Use to show the graphical representation of data.

Uses/Interactions

The output of this component is used by "End User" (e.g. Students, Universities administration, etc.).

Resources

➔ Libraries
- React JS
- Material UI
- Google Charts
- Underscore
- MUI-Datatables

➔ Database
- Firebase (by Google)

Processing

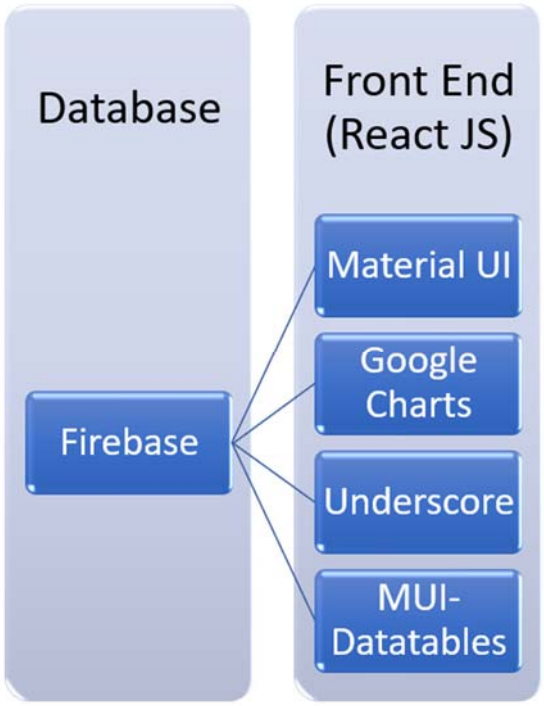
This is an easy to use web-based user interface which shows the data from the firebase database in a graphical understandable form using React JS, Google charts.

Search functionality is provided through a search bar on top of the page. It gives information about the entered tag (keyword). The job frequency in last 10 days along with tomorrow predicted the value and geographical diversity in the jobs available for that given domain.

Interface/Exports

INPUT: webpage get request
OUTPUT: graphs showing the data

Detailed Subsystem Design



5.3 CLASS DIAGRAM

5.4 ER DIAGRAM

This system doesn't have any ER diagram because we use a NoSQL database provided by Google, named firebase Realtime database

Chapter 6

IMPLEMENTATION AND TESTING

Data Extraction:

For Data Extraction using manual hardcoded parsers or scrappers, we used Scrappy, a Python framework for web crawling and scraping. This method only works for a website which is not created in JavaScript (frameworks like Angular, React JS), for sites created using JavaScript, we used a headless browser, Selenium. In order to perform Automatic scraping, we first had to extract visible website text from a given site, for this purpose we used two different generic scripts. The first generic script is for non-JavaScript created sites, usually the ones whose extension is .html, .aspx and so on. This script used a python library named BeautifulSoup, an HTML parsing library. For the second script, which extracts text from dynamic websites, we use Selenium, Headless browser. Both the scripts are generic and work on any site provided. Once the visible data is extracted, we feed it to a Deep Learning Model, which uses CRF (Conditional Random Field) on top of LSTMs to detect Named Entities. We, in particular, use Stanford NER tagger for this task.

To summarize data extraction,
Manual Scraping ->

1. Regular Site: Scrapy, Python
2. JS Dynamic Sites: Selenium, Python

Automatic Scraping ->

1. Visible Data Extraction:
    a. Regular Sites: BeautifulSoup, Python
    b. JS Dynamic Sites: Selenium, Python
2. Named Entities Extraction: Stanford NER Tagger, NLTK, Python

Time Series Forecasting:

We use a basic LSTM layer with hyperbolic tan activation to predict the future trends of a given technology or skill. , Finally, there is a Dense layer with one linear output neuron. The optimizer used was Adam and Mean Squared error was the loss function. Data was fed in terms of 128 timesteps per sample. The data was overlapping and total data of 15 years. For implementing this model, we used keras, a python deep learning high-level library.

UI Development:

Chapter 6

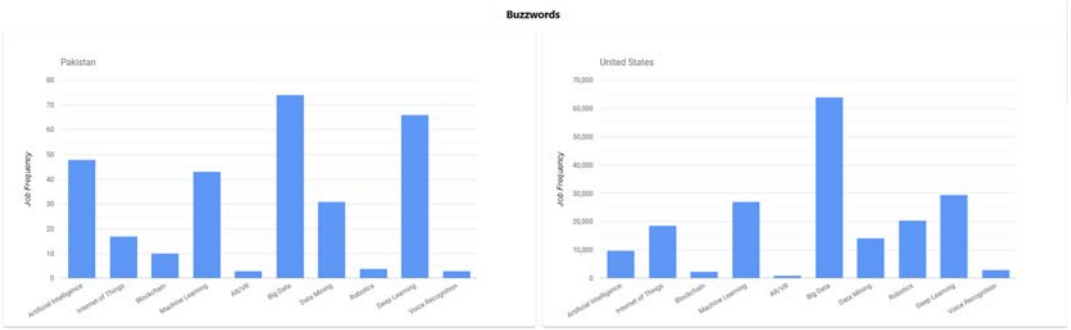
For User Interface, we created a website where all the data was visually displayed to the user. We basically used two technologies for this purpose, one is React JS, Reactive JavaScript framework and Material UI a CSS library from Google.

Each module was tested with varying values in order to ensure quality. For the prediction module, we used Mean Squared Error to test the performance of the model on 10% of the data.

RESULTS AND DISCUSSION

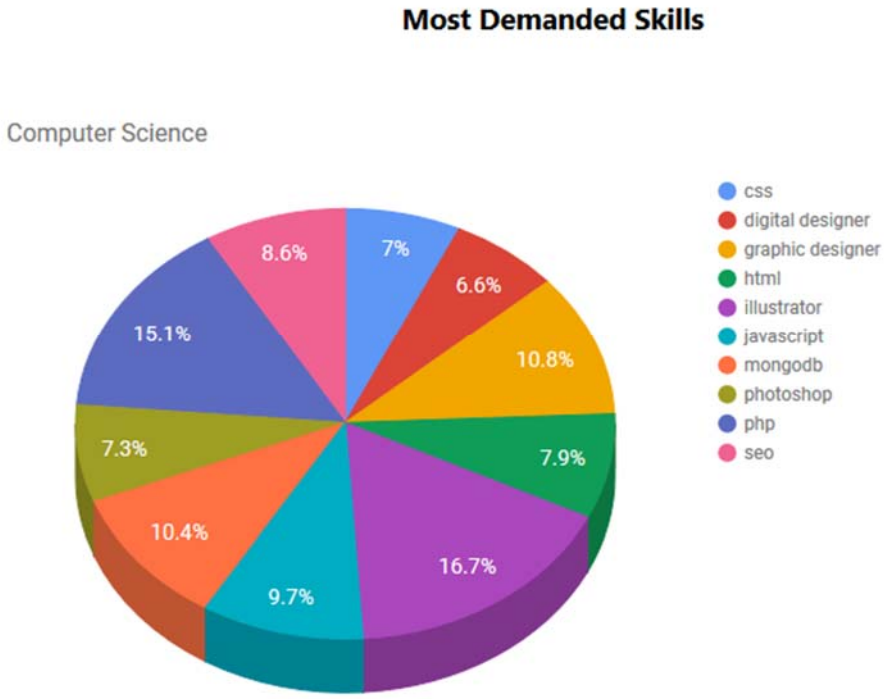
We found a number of interesting results. Firstly, let's start with a comparison of hot skill domains in Pakistan and the US.

Below is the graph which shows the current trends in both countries.



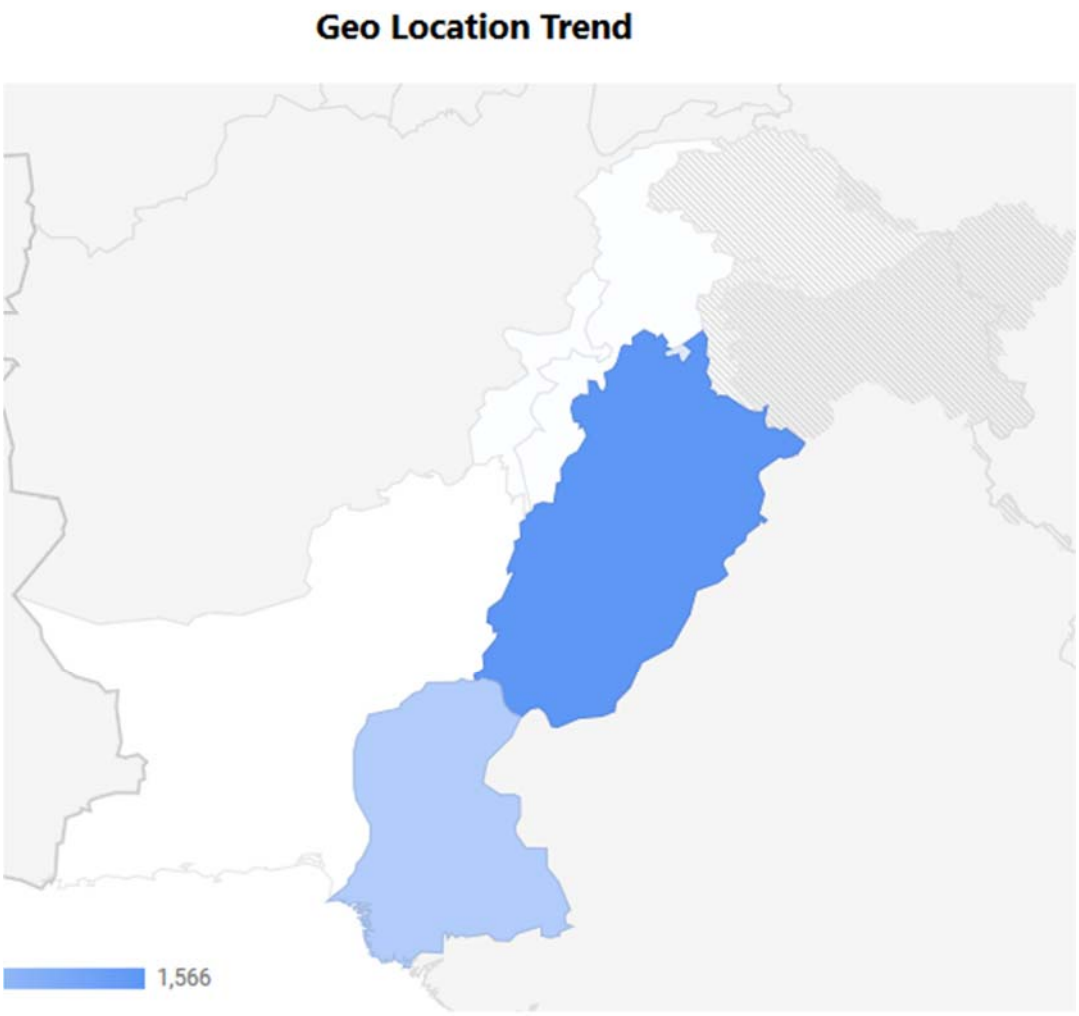
We can see that the trends are almost identical, but the magnitude varies by a lot. , Also, Pakistan is lagging behind in Robotics.

The next graph shows the percentage of jobs opened for a particular skill in a span of 30 days.



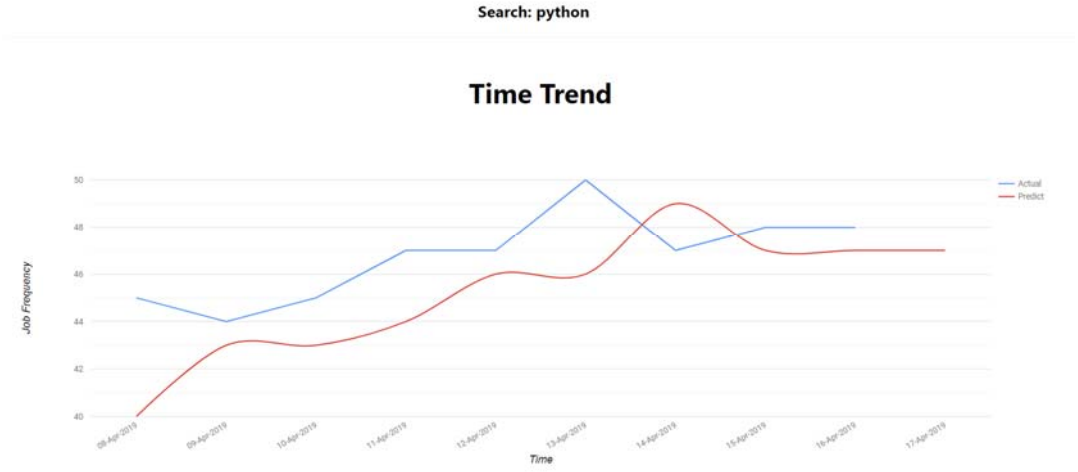
PHP is the most popular skill demanded in web development and Illustrator in Graphic Designing.

In order to show the jobs division with respect to the region, we use geo-graph.



Most of the jobs are situated in Punjab, and then in Sindh. In KPK, Baluchistan, Kashmir, and Gilgit and Baltistan the jobs situated are in very small numbers.

Following graphs show the search term (Python) job trends in the last 10 days along with over prediction



For future predictions, we used a chart to show the user the change in the trends.

There are way too many results to discuss here, but in order to simplify, I will only discuss two skills here.

One is Java for whom our System predicts drop-in popularity, and then there is Scala for which our system predicts a steady increase in the upcoming years.

To see the complete results please visit acadustia.ml.

NER Tagger Accuracy

Class	**Accuracy**
Name	98%
Location	90%
Specialization	82%

Future Prediction

Predicted change in next 4 years

Domain	6th Month	12th Month	18th Month	24th Month	30th Month	36th Month	42th Month	48th Month
Java	-1.50%	-0.50%	-1.39%	-0.67%	-0.30%	2.00%	4.19%	5.02%
Error	1.71E-04	1.94E-04	5.40E-04	4.99E-04	5.63E-04	3.38E-04	4.77E-04	3.71E-04
Cascading Style Sheets	0.00%	0.00%	-1.00%	-0.83%	-1.00%	-1.00%	-1.00%	-0.92%
Error	7.83E-04	2.49E-03	2.51E-03	4.65E-03	4.31E-03	4.14E-03	3.69E-03	3.91E-03
Scala	0.50%	0.50%	0.50%	0.50%	0.50%	0.50%	0.50%	0.50%
Error	2.72E-06	4.31E-06	5.76E-06	6.91E-06	7.80E-06	8.46E-06	9.01E-06	9.76E-06
Computer Networks	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
Error	7.02E-05	1.03E-04	7.33E-05	6.53E-05	5.55E-05	6.27E-05	5.68E-05	5.67E-05
Data Mining	-1.00%	-1.00%	-1.00%	-1.00%	-1.00%	-1.00%	-1.00%	-1.00%
Error	4.63E-05	1.22E-04	1.14E-04	1.20E-04	1.23E-04	1.25E-04	1.12E-04	1.11E-04

Chapter 8

CONCLUSION AND FUTURE WORK

In the future, we are thinking of taking it towards an implemented system with server hustings, and a properly running system from this prototype.

We are hoping to expand this system to other disciplines as well so that we could cater to the needs of the whole student community.

Moreover, we are looking forward to improving the prediction module so that we could predict the future of the next ten years.

- More accurate predictions
- The broader context of fields
- Integration with universities profile

REFERENCES

1. Brockwell, Peter J., Richard A. Davis, and Matthew V. Calder. *Introduction to time series and forecasting*. Vol. 2. New York: springer, 2002.
2. Schuster, Mike, and Kuldip K. Paliwal. "Bidirectional recurrent neural networks." *IEEE Transactions on Signal Processing* 45.11 (1997): 2673-2681.
3. Huang, Zhiheng, Wei Xu, and Kai Yu. "Bidirectional LSTM-CRF models for sequence tagging." *arXiv preprint arXiv:1508.01991* (2015).
4. Hochreiter, Sepp, and Jürgen Schmidhuber. "Long short-term memory." *Neural Computation* 9.8 (1997): 1735-1780.