Mitesh Nava
Miguel Patel
Muhammad Ashfaq

Task 1

Crawling for data:

To implement our IR-HomeFinder, we needed to find accurate data about the current real estate market. We first listed all the main real estate websites that we knew about and then started to check their robots.txt file for whether if the data was crawlable or not. We found Trulia to be the best match for our data needs and the appropriate permissions to crawl the website. We used the following tools for our crawler:

- Python 3.6
- Scrapy 1.3
- Pip
- pypiwin32

We configured scrapy's item.py file for the data fields such as:

- Price
- Address
- Bedroom
- Bathroom
- Property Link
- Square Feet

Then we wrote a spider crawler that given a link, crawls through the search results and capture the data and goes to the next page and starts over until it gets to the last page. We manually changed the links for each city. We captured a total of approximately 27,000 documents for properties in four major texas cities:

- Houston
- San Antonio
- Dallas

- Austin

Task 2

Setting up solr:

For this task of the project we

- Downloaded Solr
- Utilized Solr
- Recorded a video of the interaction with the search engine

Downloading Solr was fairly easy, along with setting it up. Since Solr does everything for us, we can simply enter a query and it will retrieve the appropriate documents using the json file that we got from task 1.

Video of utilizing solr:

We recorded a video showing the interaction with the search engine. This video tests out some of the possible queries, and results that Solr retrieves can be seen in the video. Click here for the link.

Issues with solr:

The Solr post binary wasn't working, so we had to create a core and upload it directly. The search results were pretty fast, and it worked really good with unorganized queries. It returned the whole JSON objects which we can implement in our app. Once we receive the JSON data we can format it to however we want to display it. We will be making an android app in task 3 and use solr in the backend.

Task 3

Task 3 consisted of combining what we did in tasks 1, and 2. We ending up creating an android application. However, in order to have a fully functional application there must be several individual tasks involved.

Fetching data using a separate script:

A separate python script was created which involved capturing query results utilizing solr's api. This is done by formatting the URL solr provides, and saving those results using the simplejson module. This script will not be called manually, instead there will be another python script that utilizes the bottle.py module and imports this script that is created.

Utilizing bottle.py:

We ended up using bottle.py that will simply take advantage of html requests. We then use the requests that are passed in, and give it to a function that retrieves the documents from solr. This function is in the datafetch.py script, which is the script that was discussed above. An HTTP request is sent from the android application that states the city, zip code, bedroom number, and bathroom number. The request is formatted so that the server can handle the specified request and retrieve the data from solr. The server creates a link to connect to server with the specified data input and sends it to solr. Solr returns a link to the json data which the server then opens and retrieves the data. The server then returns the data as a json string to the android app where the app handles how the data is displayed.

Android application:

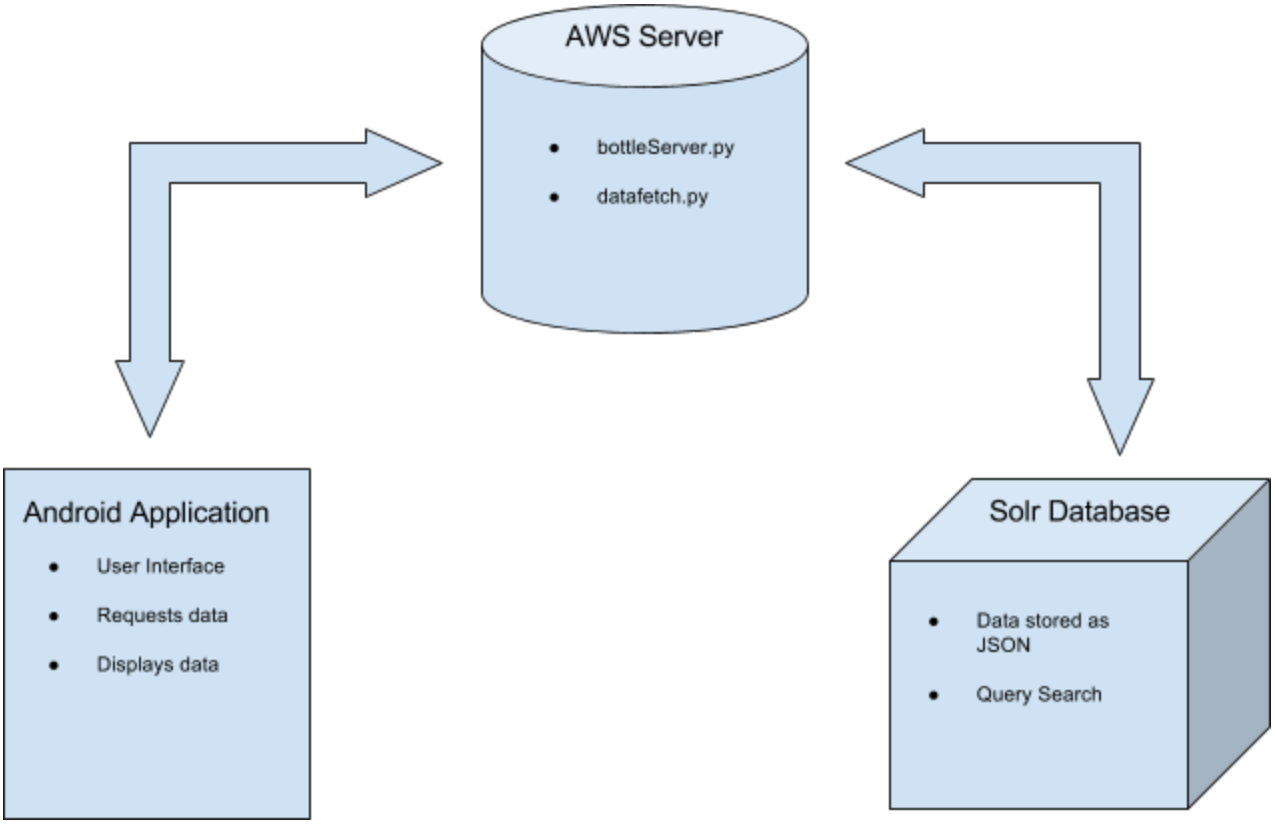
Our android application is built using android studio. We created an interface where a user enters city, zip code, bathrooms wanted in the house, and bedrooms wanted. Using this information, we send a HTML request to the server that bottle.py is running. Once this request gets made, our bottle.py application will return the appropriate documents in json format. These documents are then processed by android studio, and presented to the user in a row by row

format. In sum, with the use of this application users will be able to execute a query of their choosing, and have the appropriate results returned to them.

Server configuration:

You may be asking yourself: “Where are we running the servers?”
We utilized AWS to do as follows:

- Run solr.
- Run python script on AWS server
- Run Android application



Project Report

Introduction

The IR system we constructed consisted of retrieving information in regards to search queries focusing on cities, amount of bedrooms, amount of bathrooms, zip code, and a option to specify a price range. We wanted to make a fast IR system that retrieved accurate results. Today's applications can tend to be buggy, and return irrelevant information. However, we believe to have fixed some of those issues that we've seen in the state of the art. As a result, we expect users using our application have a less chance of frustration because of the simplicity that has been implemented.

Prior Work

As far as prior work goes, our project implemented solr, which in return implements many of the information retrieval techniques that we have discussed in class. Some of these include:

- Indexing
- Stemming/Lemmatization

Model/Algorithm/Method

Our information retrieval application is an android app. We built this application using android studio. However, building an application of this magnitude requires more than one program to successfully obtain the results that users are seeking. We split our application into parts as follows:

- Starting multiple servers utilizing AWS (amazon web services)
- Fetching data on the back end side
- Created an android application as our front end

First, we start solr on AWS, that way it is constantly running. This ensures a reliable connection so users do not have to be faced with poor connection issues. Next, we start running the

python script that uses the bottle module. This bottle module allows us to send in HTML requests that will return a string of json objects that we explain in the next step.

Secondly, fetching data from solr consisted of utilizing the link solr generates already, and modifying it depending on what the user is seeking. This is done via a python script (datafetch.py). We then get the json contents that this link points to. This is done via the simplejson module that can be installed easily. This data that is fetched, is used in the first step.

Lastly, we created an android application for our front end. This application has a search box in which users can enter a query. We use the information that we got from the first step (the json data) and output this in a nice row by row format. Users can then observe the results, and once they click on the results, they will be redirected to trulia where the information originally stemmed from. The whole idea behind creating an application geared toward housing, was to make it as simple as possible for the user. We believe we have achieved that by building this real estate application.

Results and findings: