AirBnB Listings by Reputation and Description

Talha Oz
Computational Social Science
George Mason University
Fairfax, VA, USA
+1-571-201-0575
toz@gmu.edu

ABSTRACT
Reviewing is a feedback mechanism that e-commerce sites leverage to help their customers make more informative purchase decisions on their platforms. Although the biggest online sellers such as Amazon and eBay allow their users to filter the search results by seller reputations, the leading space sharing platform AirBnB lacks this crucial feature. Even more disappointingly, AirBnB does not allow it's users to search for keywords within listing contents (descriptions). In this project, I create a demo geo-web application to meet these needs of AirBnB users. The application allows its users i) to filter the listings by review scores for six reputation categories, ii) to search in listing descriptions, and iii) to experience better visualization by adopting a different marker for each listing room type and by providing clustered-listings view. I demonstrate the application for the Washington, D.C. area by utilizing a publicly available AirBnB listings dataset.

Categories and Subject Descriptors
H.2.8 [**Database Applications**]: Data Mining

General Terms
Algorithms, Measurement, Experimentation, Human Factors.

Keywords
Reputation, geo-web mapping, AirBnB

1. INTRODUCTION
AirBnB is one of the greatest success stories of sharing economy, a website of value $25.5 billion as of November 2015 [1] where hosts provide lodging spaces and guests rent them. Hosts basically rent three types of spaces using this platform: `entire home`, `private room`, and `shared room`. In return of the quality they received during their rental, guests then leave feedback, some of which is public, to the hosts. In addition to the option of free text comments, AirBnB provides six review categories where guests can rate their experience from zero to five (in 0.5 incremental steps). This richness of feedback types is very valuable as trust is of great importance in the sharing economy.

One of the main parts, if not the main part, of listings on AirBnB website is the free-text `description` section where hosts strive for describing their property as attractive as possible. Surprisingly though, currently the website does not allow for searching.

It is unfortunate that the guests can see the listing descriptions as well as the review scores, while not being able to narrow their search exploiting this information. In this project I create a geo-web application to overcome this problem.

In this report, I first introduce AirBnB and describe the purpose of my demo geo-web application in this (*Introduction*) section. In the next (*Data*) section I then provide some of the characteristics of the dataset on which I built this demo application. The third section is about the *Design* of the application where I discuss it under two subsections as *Back End* and *Front End*. I then conclude the report with *Conclusion and Discussion* section. Tables and code snippets are added to appendices whenever found necessary.

2. DATA
2.1 Source
AirBnB (as of December 1, 2015) does not provide a publicly accessible application programming interface (API) for developers to collect information about the listings on their platform. However, enthusiastic hackers have managed to collect the listing data by implementing web scrapers (a search of 'airbnb data' in GitHub lists some).

The dataset (`Listing.csv`) being utilized in this study is retrieved from `insideairbnb.com` website and also made available in the public repository of this project. The original data source provides the date they scraped the listings, which happens to be October 3, 2015 for Washington, DC.

2.2 Descriptive Statistics
There are a total of 3723 listings in Washington, D.C area in the dataset. The very first question one might ask is the spatial distribution of these listings. Are people in Georgetown area more willing to host (list) their properties on AirBnB than those in Foggy Bottom? What neighborhoods are leading in the listings count? To be able to answer questions of this kind, I created a table (Appendix A) as well as a map (Figure 1) showing the number of listings per neighborhood.

The AirBnB listings dataset is also attribute rich, `Listing.csv` has 91 columns (Appendix A), including listing id, name, neighborhood, room type, description, latitude, longitude, host id, host listings count, number of reviews, and review scores. Other than the total review score, each listing reviewed has scores of six review categories: `accuracy`, `check-in`, `cleanliness`, `communication`, `location`, and `value`. Then one might wonder what the average review score for each category is. For all of the six categories, I found that (Figure 2) the guest satisfaction in general is very high; *value* and *cleanliness* are the lowest two with 9.32 and 9.33 respectively, and *communication* is the highest

Conference'10, Month 1–2, 2010, City, State, Country.

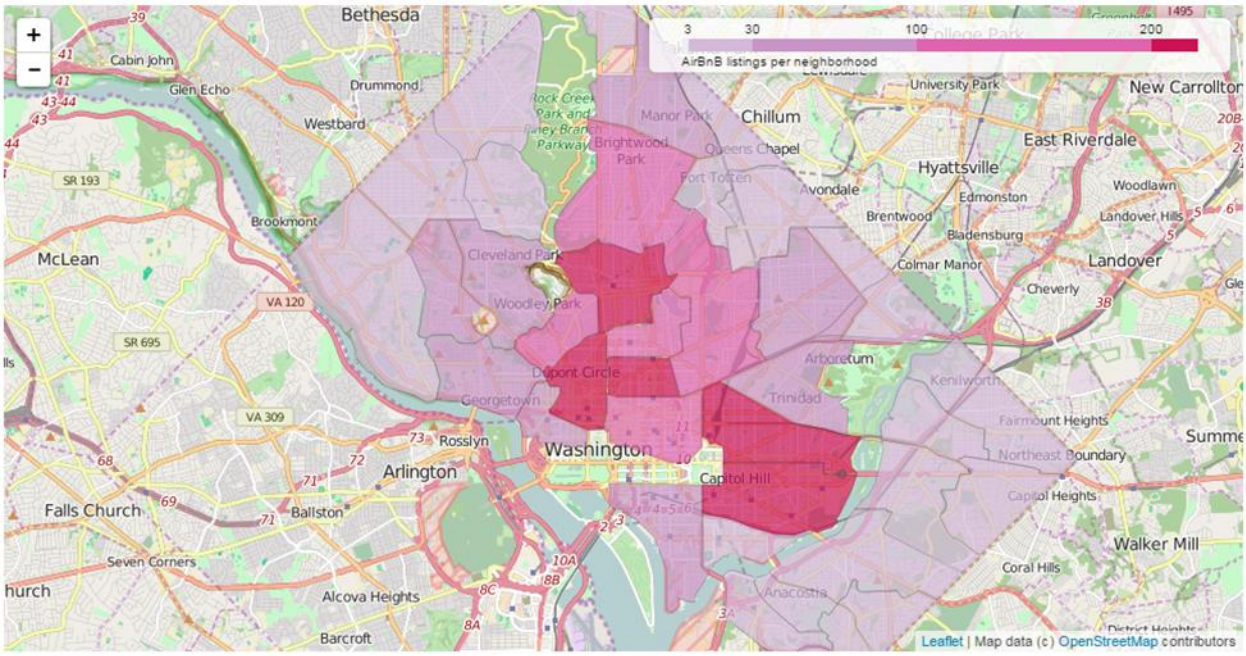

Figure 1 AirBnB listings per neighborhood

with 9.75 (One of course from these results should not interpolate that the hosts in the capital are good communicators but dirty, just as the main theme of the city, the politics itself). I should note that only 2846 listings of 3723 are reviewed at least once.

For data management operations, Python's Pandas library [7] is used, and for visualization Folium [4] and Seaborn [8] exploited.

3. DESIGN

A typical web application stack consists of a database server, a web server, a server-side web application framework, and front-end libraries (JS/CSS). A geo-web app on the other hand requires specific technologies and configuration.

3.1 Back End

On the server-side a geo-web app needs to store, operate on and communicate spatial data types effectively. First, regarding spatial data storage and operations, PostgreSQL along with its PostGIS extension allows keeping the data in various *geometry* types including *Polygon* and *Point*. Therefore, I import all the listings reviewed in the `Listing.csv` data file along with the more related columns (selected columns and rows can also be found in a file named `reviewed_listings.csv` in the repository) to my application into a PostgreSQL database (see Appendix B for the database schema), and create a new column[1] of type *geometry* to keep and operate on the listing coordinates in a single column as *points*. For full text searching I leverage text search functions and operators in PostgreSQL [6].

Second, to communicate the geospatial data better, I make us of a web server that is effective in creating responses to spatial client requests such as panning and zooming and that can handle specialized protocols such as WMS and WFS, namely GeoServer. I create a store for connecting to the database, and generate a layer (view) on top of it to be published by the server. The code snippet being used for creating the view is available in Appendix C. Given the parameters, the server configured to create and send JSONP objects over WFS when requested (by the browser).

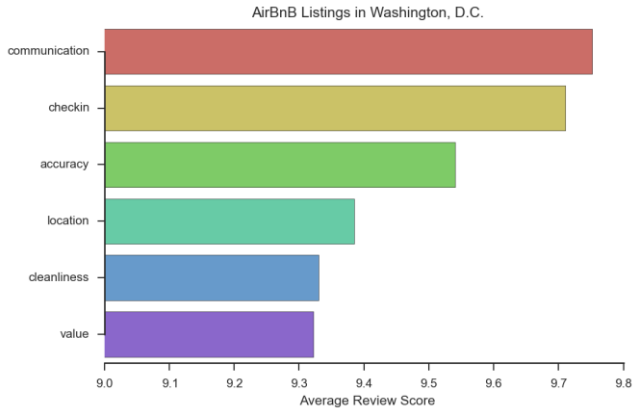

Figure 2 Average review scores of the listings in six categories

3.2 Front End

The front end of the application works in the browser and thus heavily depends on JavaScript libraries. The application utilizes Leaflet library [5] for mapping (in particular using its *geoJson*, *Icon*, and *markerClusterGroup* classes) and uses JQuery's *ajax* method [9] for asynchronous communication. In addition to these, it makes use of a rating plugin built on top of Twitter's Bootstrap library [3], namely Krajee's star-rating plugin (open sourced and available on GitHub online repository hosting service) [2]. Finally, I use three markers from the Map Icons Collection project[2] to denote the room type of the listings.

When the application is run, it basically shows a map of inquired region, along with some control tools. Since this demo focuses on the Washington, DC area, for the initial settings I set the center of the map accordingly (of latitude and longitudes: ~39, ~-77) with a zoom level enough to show the entire district but not much more.

[1] `alter table listings add column geom geometry;`
`update listings set geom = ST_GeomFromText('POINT('`
`|| longitude || ' ' || latitude || ')',4326);`

[2] https://mapicons.mapsmarker.com/

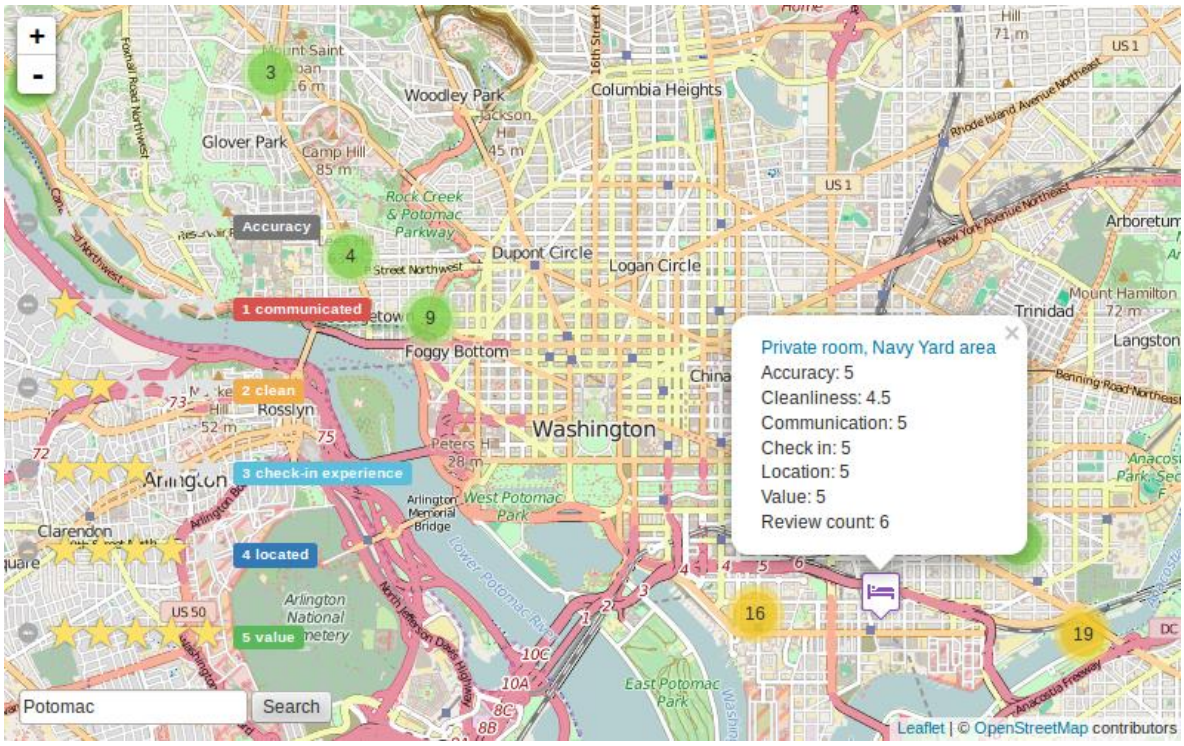

Figure 3 A screenshot of the app

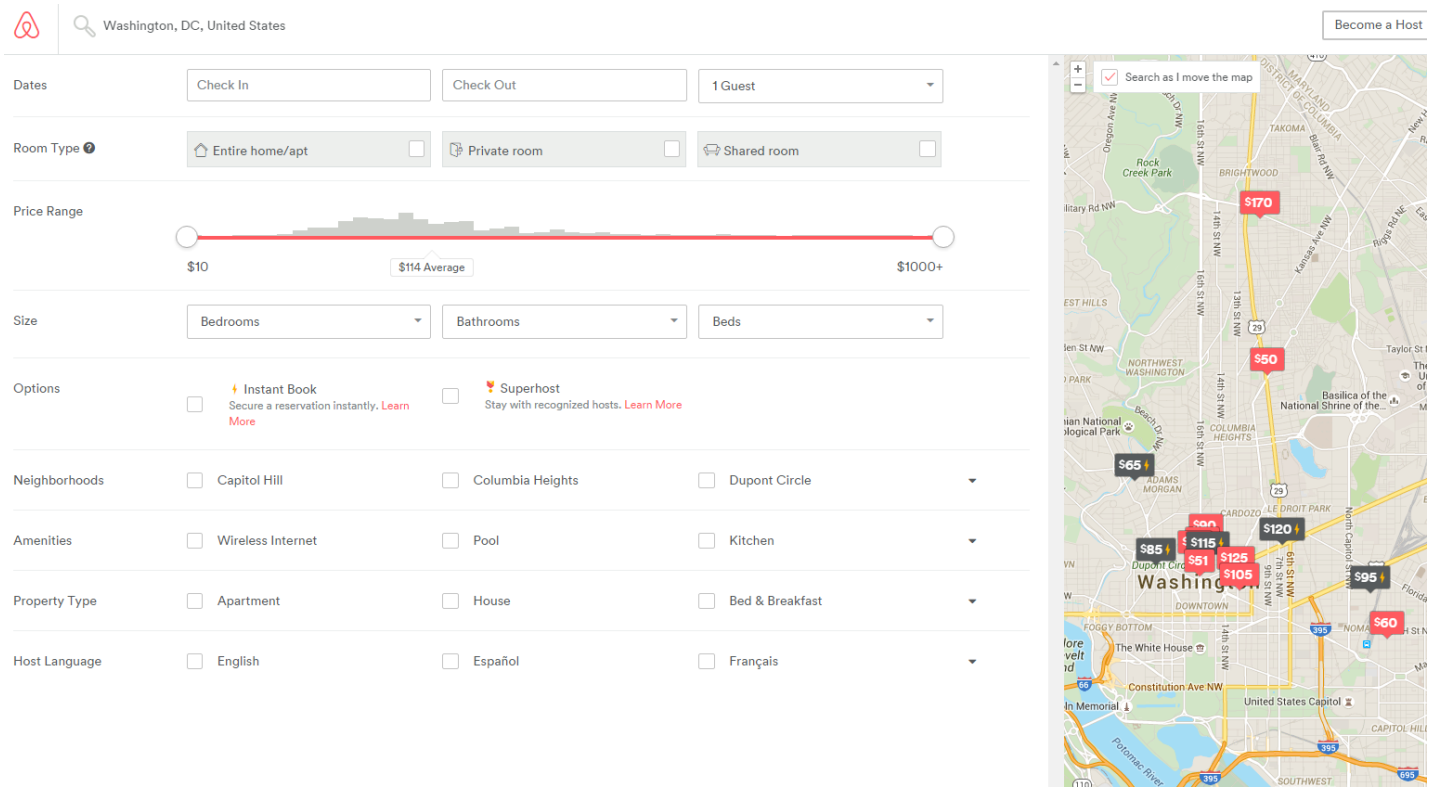

Figure 4 A screenshot of the AirBnB website showing the lack of i) keyword search, ii) review score filtering, iii) marker specialization and clustering features

4. CONCLUSION

Utilizing an effective geo-web app development stack, the demo application extends upon AirBnB website. The contributions are three-fold, the application allows its users i) to filter the listings by review scores for six reputation categories by clicking on the star ratings, ii) to search in listing descriptions by entering a key phrase into the search box, and iii) to experience better visualization by adopting a different marker for each listing room type and by providing clustered-listings views.

5. APPENDICES

5.1 Appendix A

Distribution of 3723 records over neighborhoods, top 12 listed:

neighborhood	**count**
Columbia Heights, Mt. Pleasant, Pleasant Plain...	351
Dupont Circle, Connecticut Avenue/K Street	285
Capitol Hill, Lincoln Park	242
Shaw, Logan Circle	239
Union Station, Stanton Park, Kingman Park	234
Edgewood, Bloomingdale, Truxton Circle, Eckington	194
Kalorama Heights, Adams Morgan, Lanier Heights	183
Brightwood Park, Crestwood, Petworth	140
Downtown, Chinatown, Penn Quarters, Mount	135
Howard University, Le Droit Park, Cardozo/Shaw	114
West End, Foggy Bottom, GWU	95
Georgetown, Burleith/Hillandale	87

List of the all attribute/column (91) names:

listing_url, scrape_id, last_scraped, name, summary, space, description, experiences_offered, neighborhood_overview, notes, transit, thumbnail_url, medium_url, picture_url, xl_picture_url, host_id, host_url, host_name, host_since, host_location, host_about, host_response_time, host_response_rate, host_acceptance_rate, host_is_superhost, host_thumbnail_url, host_picture_url, host_neighbourhood, host_listings_count, host_total_listings_count, host_verifications, host_has_profile_pic, host_identity_verified, street, neighbourhood, neighbourhood_cleansed, neighbourhood_group_cleansed, city, state, zipcode, market, smart_location, country_code, country, latitude, longitude, is_location_exact, property_type, room_type, accommodates, bathrooms, bedrooms, beds, bed_type, amenities, square_feet, price, weekly_price, monthly_price, security_deposit, cleaning_fee, guests_included, extra_people, minimum_nights, maximum_nights, calendar_updated, has_availability, availability_30, availability_60, availability_90, availability_365, calendar_last_scraped, number_of_reviews, first_review, last_review, review_scores_rating, review_scores_accuracy, review_scores_cleanliness, review_scores_checkin, review_scores_communication, review_scores_location, review_scores_value, requires_license, license, jurisdiction_names, instant_bookable, cancellation_policy, require_guest_profile_picture, require_guest_phone_verification, calculated_host_listings_count, reviews_per_month

5.2 Appendix B

```
CREATE TABLE listings (
    id integer PRIMARY KEY,
    host id integer,
    accuracy integer,
    cleanliness integer,
    checkin integer,
    communication integer,
    location integer,
    value integer,
    number of reviews integer,
    listing_url character varying(50),
    name character varying(50),
    neighborhood character varying(100),
    room type character varying(15),
    description character varying(1000),
    latitude double precision,
    longitude double precision);
```

5.3 Appendix C

```
SELECT
name,listing_url,room_type,cleanliness,accura
cy,communication,checkin,location,value,numbe
r_of_reviews,the_geom
FROM listings
WHERE
 (to_tsvector(description) @@
 to tsquery(regexp replace(trim('%word%'),
E'\\s+', '&', 'g')))
 and cleanliness >= cast('%cleanliness%' as
integer)
 and accuracy >= cast('%accuracy%' as
integer)
 and communication >= cast('%communication%'
as integer)
 and checkin >= cast('%checkin%' as integer)
 and location >= cast('%loc%' as integer)
 and value >= cast('%val%' as integer)
```

6. REFERENCES

[1] Airbnb, Inc. - Financial Report | Annual Revenue | Stock: *http://www.privco.com/private-company/airbnb*.

[2] Bootstrap Star Rating - © Kartik: *http://plugins.krajee.com/star-rating*

[3] Bootstrap · The world's most popular mobile-first and responsive front-end framework.: *http://getbootstrap.com/*.

[4] Folium: Python Data. Leaflet.js Maps. — Folium 0.1.2 documentation: *http://folium.readthedocs.org/en/latest/*.

[5] Leaflet — an open-source JavaScript library for interactive maps: *http://leafletjs.com/*.

[6] PostgreSQL 9.1: Text Search Functions and Operators: *http://www.postgresql.org/docs/9.1/static/functions-textsearch.html*.

[7] Python Data Analysis Library — pandas: Python Data Analysis Library: *http://pandas.pydata.org/index.html*.

[8] Seaborn: statistical data visualization — seaborn 0.6.0 documentation: *https://stanford.edu/~mwaskom/software/seaborn*

[9] jQuery.ajax() | jQuery API Documentation.