Places Insight: A Natural Language Search Engine for Places

# Project Team members

* Akhmad Rahadian Hutomo: [ahutomo2@illinois.edu](mailto:ahutomo2@illinois.edu) (team leader)
* Alexander Nestle: [nestle2@illinois.edu](mailto:nestle2@illinois.edu)
* Yang Liu: [yangl18@illinois.edu](mailto:yangl18@illinois.edu)

# Function

The goal of this project is to create a search engine implementation that will allow users to query a database of businesses, places, and parks (which will be referred to as ‘places’ within this document) using natural language attributes such as ‘child friendly’, ‘Italian food’, or ‘beautiful building’ in order to quickly receive relevant places.

|  |  |
| --- | --- |
| User will be interfacing with a keyword search that will predict the keyword based on the topics that has been generated and user will have the options to select from a pre-defined category | *Fig 1. User interface mockup* |

## Use Case

The Data-User-Service (DUS) Triangle for this project will be as follows:

*Fig 2. Use case / data-user-service triangle*

# Similar Tools

Travel and review companies, such as *Yelp* and *Trip Advisor*, generally feature a search bar on their homepage which allows users the ability to use query terms in order to retrieve relevant places. However, these sites use the query terms to find relevant reviews using a unigram model. This often provides non-relevant results, for example, the query “child friendly” will feature reviews that mention “vegan friendly”.

Our system overcomes this by using topic modeling to label places based on what is mentioned in review and then enabling the user to query the labels rather than the reviews themselves.

The team has researched some of the current open source tools and implementation that may be related and found some reference:

1. Place Search Service: <https://github.com/Attriumph/Place-Search-Service>

This implementation may be possible project to expand upon, as there is overlap with our project. The project doesn't do anything along the lines of text retrieval/analysis as it just sends the search keyword/categories directly to Google's search and displays the results, so we would either need to rewrite the entire backend in Python or rewrite the endpoints using Node.js in order to implement the topic modeling/dataset. Essentially, the project would provide us with the PHP frontend and some more functionality than our project would have otherwise.

1. My Trip Planner: <https://github.com/nhinhdao/my-trip-planner>

The solution has a way to do natural search, but there is no specialized way to rank and retrieve information with specialized place of interest model

1. Amazon Product Search Datasets: <https://github.com/QingyaoAi/Amazon-Product-Search-Datasets>

The project has sample implementation of searching into a review dataset that may provide a reference to our project

# Technology

## View

The application will be a web application. The interface will only receive user queries and display the results. The places in the results will have a URL and/or Geo location that can be used to link the user to relevant pages such as Google Maps or business web page.

* Python Flask Framework
* Bootstrap CSS Framework

## Backend

The backend will consist of a simple API that will return static templates to display results.

* Python Flask Framework
* Python library Gensim to do the topic modeling
* Python library NLTK for word tokenizer, stemming, stop words processing, etc.

## Database and Queries

Currently, the place names and addresses were scraped from Facebook travel page recommendations, which were then used to query the Google Places API to receive place information and reviews. We may also collect more place/reviews from different sources (say, yelp, Kaggle) for any upcoming iterations of projects.

Topic modeling will be used to create the ‘queryable attributes’ of the places. This will be done by running the topic modeling algorithm (say LDA algorithm) against all the collected reviews. This will create the queryable topics (place attributes). During the topic modeling process, we will need to manually interpret the topics and name them appropriately.

Once the topics are created, we will run each review through our topic model. This will tell us the topics that each review included, and the most common topics will be used to describe each place, which is then queryable by the user.

Once the places are labeled with the topics, we will use TF-IDF and inverted index to enable fast user queries. This would be done by treating each list of place topics as a document.

The storage of the database was initially designed through a simple json file. But may subject to change to NoSQL database if the volume of data increases during project implementation and enhancement.

Example of initial JSON datasets:

{

"formatted\_address": "Brooklyn, NY 11201, USA",

"geometry": {

"location": {

"lat": 40.6896147,

"lng": -73.9858984

},

"viewport": {

"south": 40.6882965197085,

"west": -73.9872249802915,

"north": 40.6909944802915,

"east": -73.98452701970848

}

},

"name": "Brooklyn",

"place\_id": "ChIJq1RGWExawokRC5VK08ax-Ds",

"reviews": [

{

"author\_name": "hisham nofal",

"author\_url": "https://www.google.com/maps/contrib/114083506813599492082/reviews",

"language": "en",

"profile\_photo\_url": "https://lh3.ggpht.com/-2hGHNr4CZDQ/AAAAAAAAAAI/AAAAAAAAAAA/CN\_PCdB3E8E/s128-c0x00000000-cc-rp-mo-ba6/photo.jpg",

"rating": 4,

"relative\_time\_description": "11 months ago",

"text": "For bus is good",

"time": 1538667162

}

],

"types": [

"transit\_station",

"point\_of\_interest",

"establishment"

],

"url": "https://maps.google.com/?cid=4321399309968512267",

"html\_attributions": []

}

## Data Cleansing

A few data cleansing tasks has been done to the initial dataset to ensure there are no duplicates of places/ business entries, the place has a certain threshold of number of reviews and also count of the words inside the reviews

# Tasks

## Completed

* Create the dataset by scraping names from Facebook Travel Pages and querying Google Places API
* Implement data cleansing

## October

* Developing topic modeling algorithm like LDA
* Manually label topics
* Function to compare each review for each place against the topic model and save the resulting topics to place
* Create inverted index for fast queries
* Evaluate and implement TF-IDF ranking function
* Develop Backend implementation framework

## November

* Develop Front end view (Python Flask Templates)
* Create System Documentation

# Potential Problems

## Application Stack Restrictions

The assignment needs to be uploaded to a public repository for the TAs and professor to access. The team assuming they will both look at the code and run the project on their local system. We will need to account for this during develop and ensure that if we choose a web application that it is easily run locally.

## Database File Size

The JSON database file could become too large for memory. We do not foresee this happening as the test database I created contain around 2,000 places and 10,000 reviews and it was only 7.5 MB.