tiaplagata / **capstone-project**

My final Data Science project at Flatiron School! Using Natural Language Processing to build a classification model that predicts where you should travel based on text saying what you want to do on vacation. Includes a DASH app with an interactive interface to try the model out for yourself! Data scraped from TripAdvisor.

GPL-3.0 License

☆ **0** stars **0** forks

☆ Star Unwatch ▾

<> **Code** Issues Pull requests Actions Projects Wiki Security

main ▾ ···

tiaplagata pushing new edits … 1 minute ago **29**

View code

README.md

The Destination Dictionary

The Destination Dictionary

Find your perfect destination
in just a few words.

Navigation

- Project Overview
- EDA
- Model Analysis
- The Final Product
- Future Work

Important Links

- Slideshow Presentation
- Non-Technical Video Presentation
- Jupyter Notebook with Data Collection
- Jupyter Notebook with Exploratory Data Analysis (EDA) and Modeling
- Dash App Repo

Project Overview

The COVID-19 pandemic has severely affected the travel industry. International travel has been impacted, and in turn travel companies and travel websites have lost much of their engagement.

However, with the development of new vaccines for the virus, there is hope on the horizon for international travel and a time where life is somewhat back to normal. In order to increase engagement in the travel industry and increase excitement about travel opportunities, the Destination Dictionary was born!

The Destination Dictionary is a data product that allows future travelers to get a prediction for their perfect destination with the input of just a few words. Trained on over 28,000 unique text data points, the Destination Dictionary is able to predict a destination from 12 different popular cities with 81% accuracy based on text input of activities you want to do while on vacation.

Methodology & Data Used

This project utilized data from 12 top cities from TripAdvisor's list of Traveler's Choice destinations for Popular World Destinations 2020, which can be found via this link. The dataset was compiled by scraping the titles from Tripadvisor 'attractions' for each of the 12 cities. The final dataset included over 28,000 unique text values.

EDA

The Exploratory Data Analysis for this project was mainly devoted to exploring some cleaning and preprocessing techniques for the text data using Natural Language Processing. I also investigated different vectorization strategies for the data and looked into specific words/phrases that needed to be cleaned/removed from the dataset.

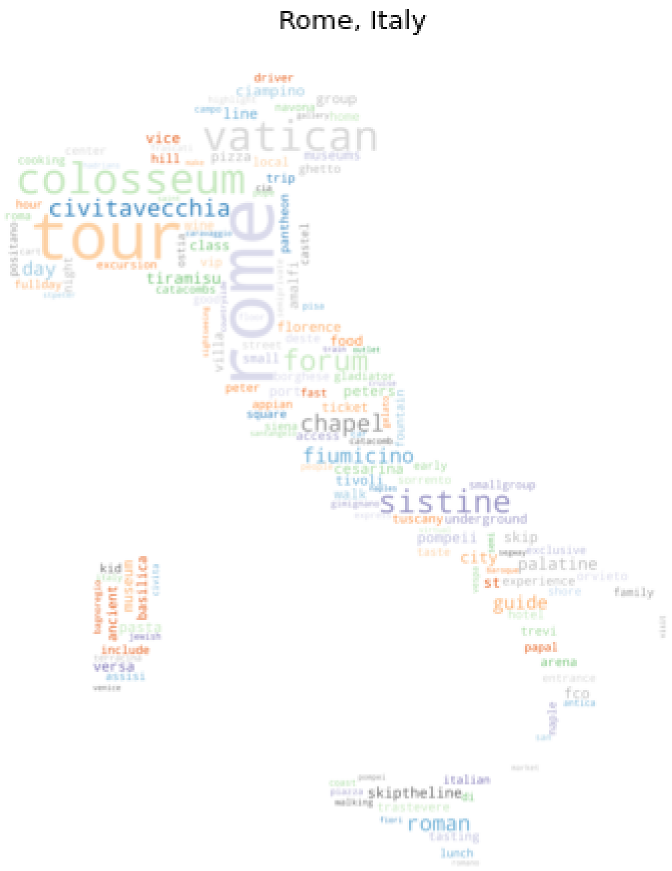
Findings

In order to classify the text data into different classes, I experimented with 3 different vectorization strategies:

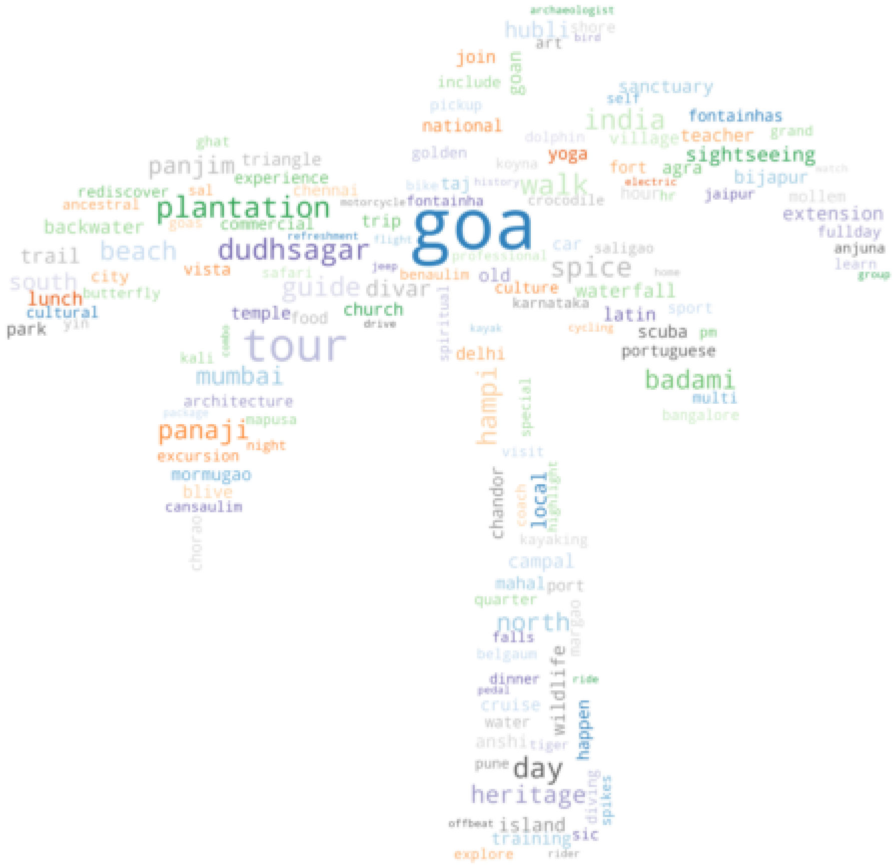
- Count Vectorization
- TF-IDF Vectorization
- Count Vectorization using Bi-Grams

After exploring the document term matrices, word clouds, and most frequent words from each of the 3 vectorization techniques, I concluded that the best vectorization strategy is to use TF-IDF vectorization. This strategy is often used to classify news articles into their correct topics, which is a similar use-case to the Destination Dictionary. The purpose here is to classify the text to the correct city, therefore TF-IDF vectorization provides the appropriate ratio to reflect how important a word is to each specific city.

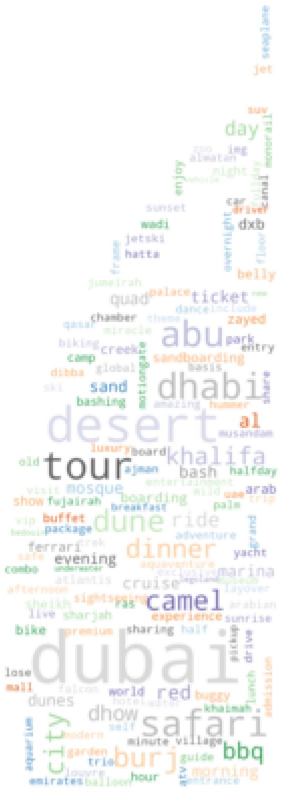
Sample Word Clouds with TF-IDF Vectorization



Goa, India



Dubai, United Arab Emirates



Recommendation for Modeling

Based on the results of the EDA, I would recommend removing the following words from the corpora/text data:

- City names (i.e. 'Paris', 'Rome', 'Sicily', etc)
    - Most of the attractions are listed with the city name included (ex. Paris Champagne Tasting), however when the model gets new data, it will not have the city name. For this reason, it is best to train the model without the city names to get scores that are more indicative to its performance for its use-case.
- The words 'private', 'airport', and 'transfer'
    - Most of the cities have 'Private airport transfer' as an attraction listed in their corpora. Since this does not tell us anything about the city, we should remove these data points/words before training the model.

Model Analysis

My final model is a Multinomial Naive Bayes classifier, which can predict a destination with 81% accuracy and an 82% F1 score. The text data put into this model is not lemmatized, but is lowercased with stopwords removed and city names removed.

Model Fit & Score

I used accuracy and F1 score to score this model. Since there are 12 classes, I want to model to be accurate, however, F1 score is also important to consider since there is some class imbalance in the dataset and to account for the model's false positives and false negatives.

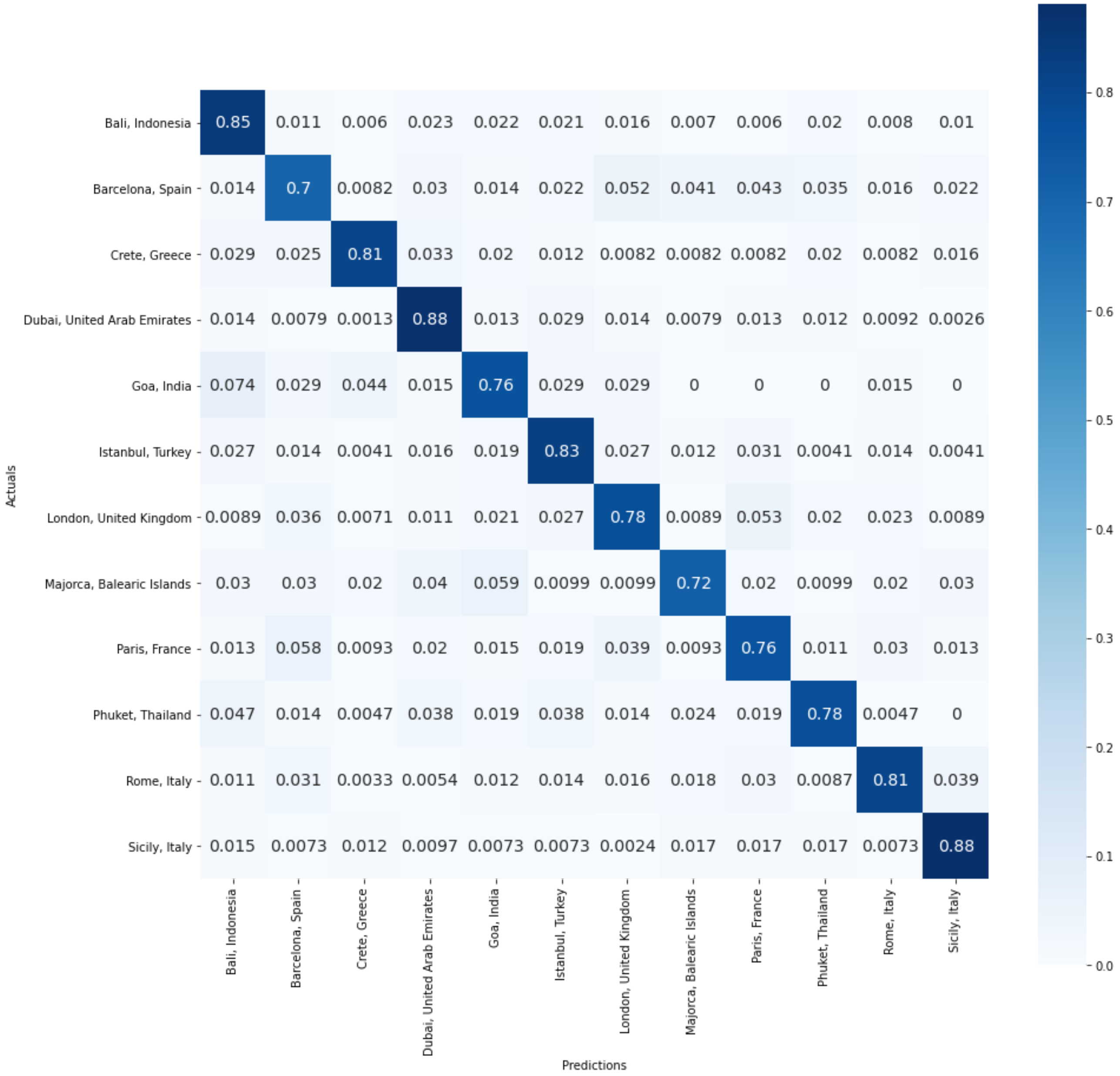
The final model had the following training and testing accuracy and F1 scores:

- Testing Accuracy Score 0.81 | F1 Score 0.82
- Training Accuracy Score 0.86 | F1 Score 0.86

Looking at the above scores for both accuracy and F1, we can conclude that the model is a tiny bit overfit, but overall very accurate, especially considering that there are 12 classes.

Confusion Matrix

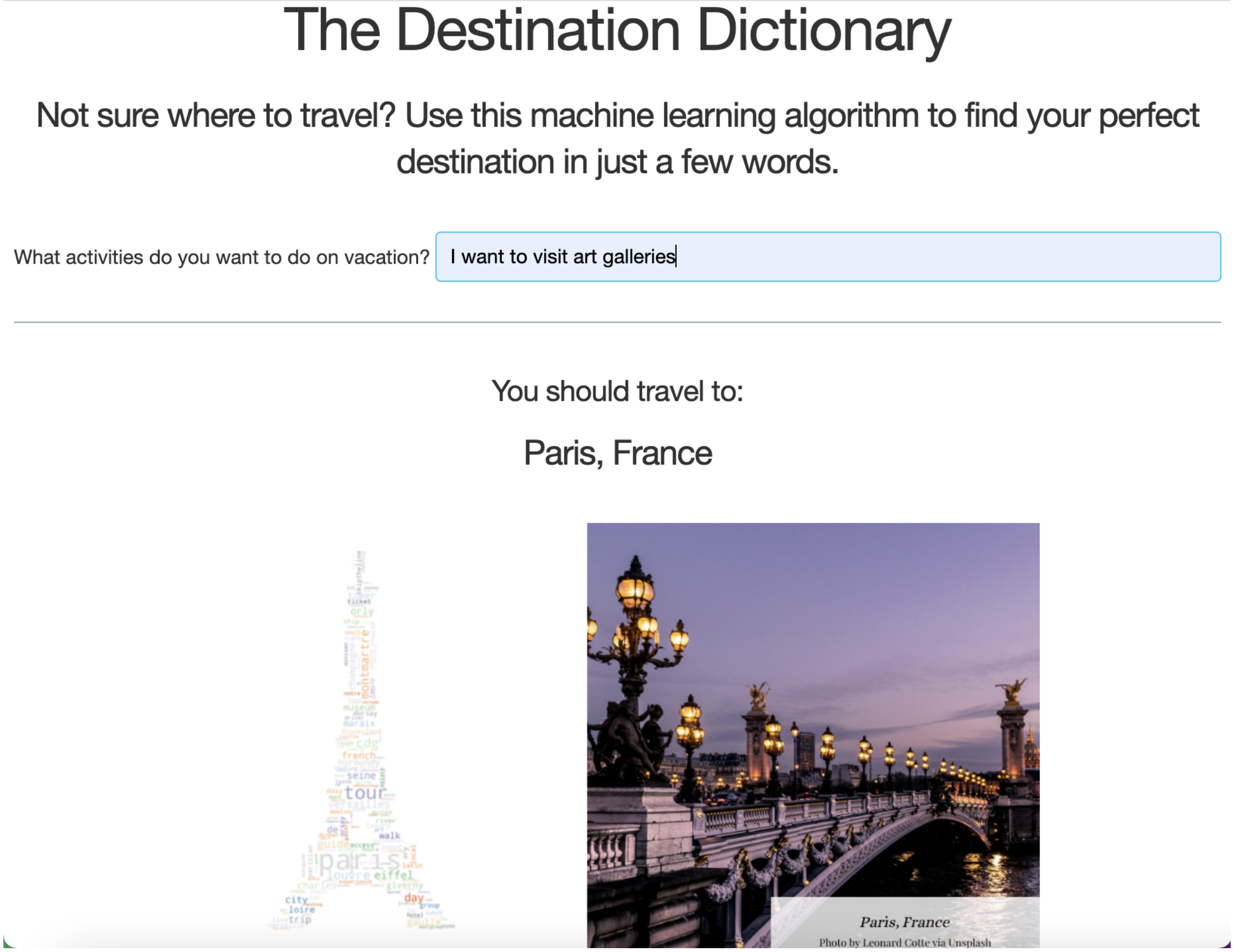
The final model's confusion matrix from the test set is depicted below.



	Bali, Indonesia	Barcelona, Spain	Crete, Greece	Dubai, United Arab Emirates	Goa, India	Istanbul, Turkey	London, United Kingdom	Majorca, Balearic Islands	Paris, France	Phuket, Thailand	Rome, Italy	Sicily, Italy
Bali, Indonesia	0.85	0.011	0.006	0.023	0.022	0.021	0.016	0.007	0.006	0.02	0.008	0.01
Barcelona, Spain	0.014	0.7	0.0082	0.03	0.014	0.022	0.052	0.041	0.043	0.035	0.016	0.022
Crete, Greece	0.029	0.025	0.81	0.033	0.02	0.012	0.0082	0.0082	0.0082	0.02	0.0082	0.016
Dubai, United Arab Emirates	0.014	0.0079	0.0013	0.88	0.013	0.029	0.014	0.0079	0.013	0.012	0.0092	0.0026
Goa, India	0.074	0.029	0.044	0.015	0.76	0.029	0.029	0	0	0	0.015	0
Istanbul, Turkey	0.027	0.014	0.0041	0.016	0.019	0.83	0.027	0.012	0.031	0.0041	0.014	0.0041
London, United Kingdom	0.0089	0.036	0.0071	0.011	0.021	0.027	0.78	0.0089	0.053	0.02	0.023	0.0089
Majorca, Balearic Islands	0.03	0.03	0.02	0.04	0.059	0.0099	0.0099	0.72	0.02	0.0099	0.02	0.03
Paris, France	0.013	0.058	0.0093	0.02	0.015	0.019	0.039	0.0093	0.76	0.011	0.03	0.013
Phuket, Thailand	0.047	0.014	0.0047	0.038	0.019	0.038	0.014	0.024	0.019	0.78	0.0047	0
Rome, Italy	0.011	0.031	0.0033	0.0054	0.012	0.014	0.016	0.018	0.03	0.0087	0.81	0.039
Sicily, Italy	0.015	0.0073	0.012	0.0097	0.0073	0.0073	0.0024	0.017	0.017	0.017	0.0073	0.88

The Final Product

The Destination Dictionary final product is a dash app hosted on my local machine (soon to be hosted on Heroku as well!). Check out a screenshot below, or see my non-technical walkthrough to see it in action!



The Destination Dictionary

Not sure where to travel? Use this machine learning algorithm to find your perfect destination in just a few words.

What activities do you want to do on vacation? I want to visit art galleries

You should travel to:

Paris, France

See the full dash app repo here

Recommendations for Use in the Travel Industry

- Integrate the Destination Dictionary technology into pages where Top Destination lists are published to drive engagement with future travelers and drive traffic to affiliate links
- Use the Destination Dictionary technology paired with a chatbot on travel websites to act as a virtual travel agent
- Offer paid sponsorship of the 'default' city-- ex. Tourism Board of Bali can pay be the first recommended city when you open the page

Future Work

If I had time to explore further, I would investigate the following:

- Continue tweaking the cleaning/preprocessing steps to improve model recall score

- Train the model on more classes/cities to include the entire 25 destination list
- Train a Deep NLP model on the dataset using LSTMs and GRUs
- Improve Dash App

Thank You!

I hope this project inspires your future travels! Please contact me with any questions:

- tiaplagata@gmail.com
- https://github.com/tiaplagata
- https://www.linkedin.com/in/tiaplagata/

Releases

No releases published
Create a new release

Packages

No packages published
Publish your first package

Languages

● **Jupyter Notebook** 100.0%