Finding Sister City

Sainag Shetty
North Carolina State University
sgshetty@ncsu.edu

Sachin Kumar
North Carolina State University
skumar26@ncsu.edu

Troy Marchesseault
North Carolina State University
tmarche@ncsu.edu

Ray Black
North Carolina State University
rwblack@ncsu.edu

Lisa Wu
North Carolina State University
jwu7@ncsu.edu

1. INTRODUCTION

Today, to develop trade and tourism, people want to know which cities have strong connection. People generally consider two cities with strong connection as sister cities. Therefore, we try to develop a web application to find a city's sister cities. Users can input a city to retrieve its sister cities and then input an arbitrary keyword to rank those sister cities. In the following part, we will state how we frame the problem, how we implement the project, what the limitations are and what we can do to improve the project.

1.1 Problem Statement

Sister Cities Policy defines sister city as a form of legal or social agreement between towns, cities, counties, oblasts, prefectures, provinces, regions, states and even countries on geographically and politically distinct areas to promote cultural and commercial ties.[1] To quantitatively compare two cities, we define sister cities as cities are similar in user-specified criteria. There are two ways to define criteria. The criteria could be the categories usually used to describe a city, such as demographics and economy. Input keywords could also be used to compare cities, such as "best coffee", to have broader criteria.

To limit scope, we only involve cities in US. We found the intersection of city lists from various source to have the complete data for each city. The final list contains 15718 cities.

To decrease the complexity of the unstructured analytics phase, for each input city, we return top 5 sister cities the most similar. We choose 5 categories as default putting in the database. Each category has multiple variables to measure the similarity. For example, we measure the median and mean income in economy, population and marital status in demographics. However, user could input arbitrary keywords to the system for ranking, which increases the complexity.

The default five categories are climate, demographics, economy, political views and proximity. Each category is a dimension used to compare similarities. A user could assign weight to each category to evaluate sister cities according to the user's preference. And then we rank sister cities according to the relevance to the input keywords.

2. LITERATURE REVIEW

2.1 Structured Approach

The first step of the analysis is the comparison of the structured data for each city. Given an input city, we wanted to find the city with the most similar metrics to that city. Because we had so many distinct criteria, and we were trying to find a similar solution rather than an optimal one, we elected to use KNN rather than TOPSIS or some other multi-criteria decision making algorithm. KNN allows us to compare the similarity of cities across each input dimension rather than projecting the data onto a lower dimensionality before making the comparison. Thus, it is easier to determine how our decisions are made in this phase of the analysis because each field in the output cities can be manually inspected.

To prepare our data for KNN, we had to normalize each dimension so that it could be compared on the same scale without unintentional weight differences. Normalization was performed with a simple min-max formula which rescales the data to the range [0,1], where 0 is the value once held by the minimum in that column and 1 is the value once held by the maximum of the column. This was done with the following equation:

$$X_i = \frac{max(X) - x_i}{max(X) - min(X)} \forall X_i \in X$$

Figure 1: Normalization formula

After normalization, we applied weights to each category. The weights are given by broad category (proximity, climate, economy, politics, and demographics) and are then applied to each dimension as a scalar. The has the effect of increasing the importance of high weight categories by preferring neighbors that are closer in that dimension. Once the weights have been applied to each data column, the scikit-learn KNN algorithm is performed. We specifically use the KD-Tree method of selecting neighbors due to the size of the input set. This algorithm calculates which k neighbors have the shortest euclidean distance (in arbitrarily high dimensional space) from the input city. Finally, the 5 closest cities are passed to the next phase of analysis which uses unstructured data to rank the 5 cities by some additional user-provided keyword.

2.2 Unstructured Approach

First step was getting news articles for unstructured text analysis. So for a provided query input and list of cities de-

termined in previous step, our script searched Google news for those querywords and after scraping those URLs from Google news page, crawled those URLs to get five relevant news articles for every city. Now after getting those articles we did need to analyze those news articles and determine its relevance to the query words. So we performed Ranking and scoring news articles using Dual Embedding Space-Model[1]

So we followed the algorithm proposed in the paper as follows:

- We trained a Word2Vec embedding model on a corpus of 1,50,000 news articles while retaining both the input and the output projections,allowing us to leverage both the embedding spaces to derive richer distributional relationships.
- During ranking we mapped the query words into the input space and the document words into the output space,and computed a query-document relevance score by aggregating the cosine similarities across all the query-document word pairs.
- A novel Dual Embedding Space Model, with one embedding for query words and a separate embedding for document words.
- Document ranking feature based on comparing all the query words with all the document words, which is equivalent to comparing each query word to a centroid of the document word embeddings.

$$DESM_{IN-OUT}(Q,D) = \frac{1}{|Q|}\sum_{q_i \in Q}\frac{q_{IN,i}^T \overline{D_{OUT}}}{\|q_{IN,i}\|\|\overline{D_{OUT}}\|}$$

Figure 2: Ranking formula

Where Q is Query,vector and D is document vector

- Embeddings-based approach is prone to false positives, retrieving documents that are only loosely related to the query. But we had solved this problem effectively by ranking based on a linear mixture of the DESM(Dual embedding space model) and the word counting features

3. METHODOLOGY

3.1 Gathering Data

Structured data for the KNN selection phase was gathered both by web scraping and by download from the US Census Bureau. For web scraping, we used BeautifulSoup as well as Scrapy. The scripts for collecting this data can be found within our submitted codebase. Once the raw data had been collected, we extracted the relevant columns and collated them into a single csv file. Notably, we included only cities for which we had complete data, which amounted to 15,718 alternatives. The cities which were not included were usually of much smaller populations.

3.2 Process

The user can start the process by providing an input city (with weights) for which we will determine sister cities. The city must be provided with the exact format which is in our

CATEGORY	DATA POINTS	DATA SOURCE	DATA FORMAT
Climate	Average Min Temp Average Max temp Average Temp Precipitation	http:// www.areavibes.com/	Web Scraping
Demographics	Population Marital Status Educational Attainment Native/Foreign Born	US Census Data	csv
Economy	Median Income Mean Income	US Census Data	csv
Politics	% Dem votes % GOP votes	http://www.townhall.com	csv
Proximity	Latitude Longitude Distance (calculated metric)	https://simplemaps.com/data/us-cities https://developers.google.com/maps/	csv and API calls

Figure 3: Data Model

list, so we use an autocomplete feature to ensure the correct city is found. Weights can be in the range [0,100] where weights of 0 ignore the category and 100 gives max priority.

The application then presents the user with a list of 5 cities most closely matching the input parameters. Included in the display are the structured data for each column, which the user can inspect more closely.

Next, the user can input a keyword string that will rank the cities based on their relevance to this keyword string. The user is presented with a re-ordered list including all cities that had news articles relevant to the input string.

3.3 Technology Stack

- Python:

  Python is a high level programming language. It was used to code entire application,due to the various libraries available with python support like scikit-lean, word2vec etc.

- Bootstrap

  Bootstrap is a free front-end framework for faster and easier web development. Bootstrap includes HTML and CSS based design templates for typography, forms, buttons, tables, navigation, modals, image carousels and many other, as well as optional JavaScript plugins

- Django:

  Django is a python based framework which follows the Model-view-controller pattern. It was used for web application integration with python backend, due to its simpler architecture providing seamless integration of views and python based backend logic.

- SQLite:

  SQLite is a relational database management system embedded into end program. It was used for saving Django's session variables and session information, due to its simpler design and easier integration with Django.

- Scrapy:

  Scrapy is a web crawling framework written in python. It was used for web scraping of information for various attributes data for building weighted criteria model, because it is really fast with its various selector methods to parse a page and extract the relevant piece of information in a much faster and efficient manner.

- Google News:

  Google News is a news aggregator that selects news from thousands of news websites. It was used for scraping news URLs for query search by user and then further crawling those URLs to get news articles. We used it because it has really good collection of articles from various sources which addressed our need for a news source that provides news for different cities across various topics.

- Google Maps API:

  It is an API for Google's web - mapping service. For getting latitude and longitude of certain cities for which we didn't had same data collected from other sources. We used it due to its extensive coverage of all the places and cities, which provided missing location data for our project.

- Scikit learn:

  It is a machine learning library for python, with api's for various machine learning algorithms. It was used for getting most relevant cities as per weights assigned to different attributes by users and also for using K-nearest neighbors algorithm. We used this library due to its fit for our problem statement of locating cities with nearest weighted scores to home city.

- Word2vec:

  It is a group of related models to produce word embeddings. It was used for building embeddings based model to get embeddings for words in document to generate its centroid, and also for query words to generate its embedding ,so that its proximity to centroid of document can be determined. We used this library as getting embeddings was part of our algorithm to determine relevance of a city to query keywords.

- Gensim:

  It is a wrapper library for loading various embedding based models and also a good library for vector space modeling and topic modeling toolkit. It was used for loading Word2vec model and generating Tensorflow vectors for visualizing embeddings. We used due to its easy to use API's to inter-operate between various Tensorflow or embedding based models implementations.

- Tensorflow:

  It is open source library for machine learning and various other neural networks based implementations. It was used for building word2vec model and generating Tensorflow vectors for visualizing embeddings and also for using it as back-end for Word2vec model. We used it because of its easier to use implementation and support for Word2vec vectors generation and visualization.

4. RESULTS

The project is a web based application which integrates the webpage and the backend. The webpage was designed using Bootstrap. It allows the user to enter the input city along with weights for the categories in the form of sliders for the first stage of results. The inputs are sent to the backend using a POST request on the submit action. Then it allows user to provides option to provide a search term to rank cities again as per its relevance to that query.

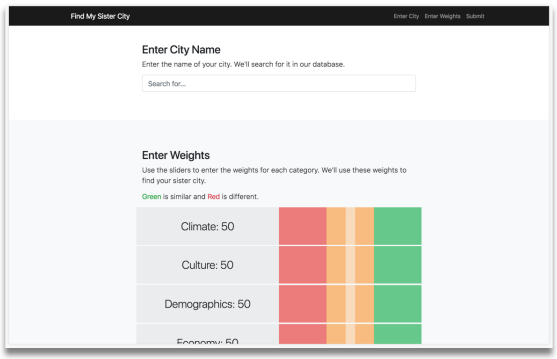

Figure 4: Landing Page

The backend of the project was built in Python. It was divided into 2 stages:

1. Structured Analysis:

   In this stage, the inputs received from the user i.e. the input city and category weights, were applied to the KNN model and a list of 5 cities were returned which matched the criteria set by the user. This list of cities along with the home city were displayed to the user, along the the values for all the respective attributes and its sub-attributes values for which user selected weights on the first page.

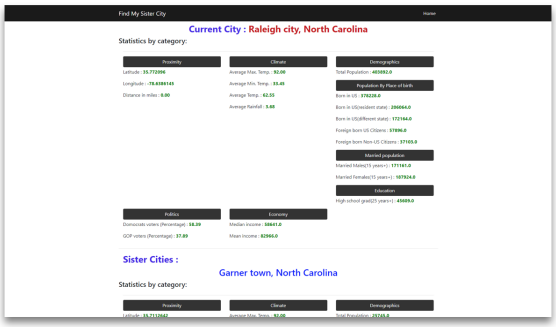

Figure 5: Result after Structured Analysis

2. Unstructured Analysis:

   After the user received a list of 5 relevant cities, the application allowed the user to enter a query value to further rank those five cities as per relevancy to that query provided. This query term is then used to fetch relevant news articles which were scraped from Google News in real-time. Then using the trained Word2Vec model, the list of 5 cities were ranked according to the scores calculated as per dual embedding space model implemented using word2vec embeddings.

   For the validation of the unstructured analysis, we didn't have judged validation sets to determine its relevance so we manually sampled articles for their rele-

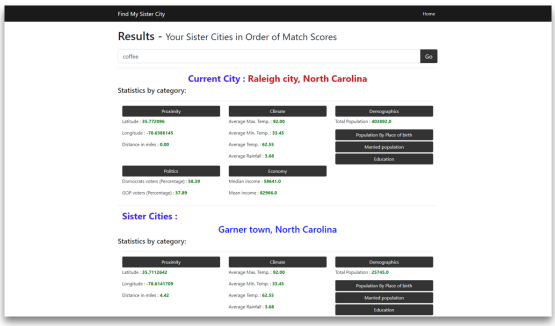

Figure 6: User enters refinement phrase

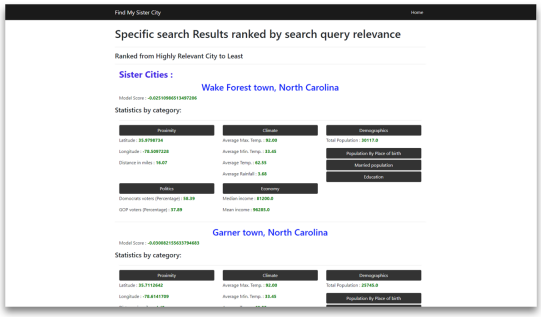

Figure 7: Result after Unstructured Analysis

vance. For word2vec embeddings we sampled embeddings to see their similarity scores. The visualization of the embeddings using t-sne which were generated as:

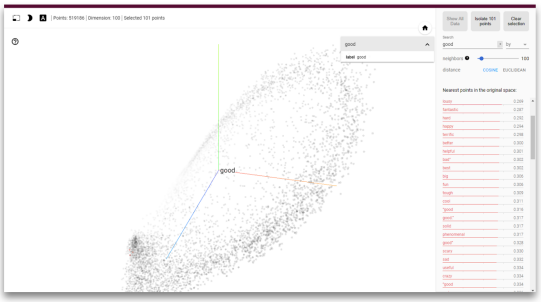

Figure 8: Validation

5. LIMITATIONS

The project, as is, has a number of limitations associated with it. These limitations range from the scope, data cleaned for analysis, news articles used for unstructured analysis (UA), the effectiveness of the UA, the design of the UA input, and the problem statement itself. This section addresses and details these limitations. The following section provides possible solutions to these limitations.

- **Scope:**
  - Limited to US cities:

    First of all, the scope of the project is limited to only US cities. Sister cities, as often established, reach all across the world.
  - Limited to Cities with Complete Data:

    Additionally, within the US, only cities with complete data were used for the analysis (about 15,000) total. This excluded many smaller towns and incorporated areas from being considered within our analysis.

- **Data for Analysis:**

  The data available to analyze the cities was limited in and of itself. For example, we originally wanted to include a "Culture" category that was based off of religion, ethnicity, and language. However, not all of the relevant Census categories (e.g. language) were cleaned and ready for use, and not all useful categories, such as religion and ethnicity, were available.

- **News Articles Available for Unstructured Analysis:**

  As is, the application only searches through articles available on Google News. This precludes many potentially relevant articles from being considered. The lack of available articles greatly restricts the unstructured analysis (UA) portion of the application, which currently does not take a city into consideration unless it can at least one article about it.

- **Effectiveness of Unstructured Analysis:**

  The current version of the UA in the application, for example, does not perform well at differentiating between "best" and "worst" in providing output. Rather, it seems, in determining relevance to a user's input, it primarily takes into consideration how often the modified word, e.g. "parks," appears in the gathered news articles.

- **Design of Unstructured Analysis Input:**

  At present, there is no limit to the variety of input that the user can give to the UA portion of the application. The user can potentially enter any string, e.g. "Who won the Super bowl in 1988?" and the application will still attempt to sort it's final list of five cities in accordance with this input string. This is clearly not desirable, and it would be best to place limitations on exactly what the user is able to enter.

- **What is a Sister City**

  As it stands, the application is limited by the problem statement itself - Identifying a city similar in aspects to a city provided by the user. The application is not user-focused, and does not take any input about the user herself. Rather, the primary focus is upon the input city. The next section details how this and other limitations might be fixed.

6. FUTURE SCOPE

This section addresses the changes that could be made to the application in order to overcome its limitations:

- **Increase Scope:**

  Moving forward, it would be ideal, however, not essential, to increase the scope of the application to reach beyond the United States. Areas where a large amount of information is available, such as Europe, would be a good place to start.

- **Acquire Additional Data:**

  - Clean Additional Census Data:

    Some useful Census data, such as language, is not cleaned or integrated into the current application. Readying this data for use would be best for future implementations of the program.

  - Find Additional Data Sources:

    We were unable to locate data sources for certain information that we thought would be useful. Features such as religion and ethnicity could be found and extracted from other sources and added to the data set.

- **Search a Larger Space for News Articles:**

  Current searches for news articles only run through those indexed by Google News. 50-States.com offers links to city newspapers from all across the United States (over 3,300 total) and could be used to supplement those from Google News. The addition of more searchable articles would allow for better comparisons across cities for the UA(Unstructured Analysis) portion of the application.

- **Building Rules based system for Specificity in Context:**

  The current UA(Unstructured Analysis) is difficult to explain and, in practice, not as effective as a rules-based system might be. The current implementation calculates and compares cosine similarities for centroids of the articles to the user's query input. While this is effective for finding relevant articles, it is not as good at scoring them. The current system could be improved upon, retaining the method for finding relevant articles; however, scoring articles also with a rules-based system to get specificity in context.

  Example New System:

  Assume the user enters the string "Best parks" Relevant articles are obtained using the original method and are broken down by sentence. We drop any sentences that don't include the base part of the search query (i.e. "park"). We then increase the score of the article for sentences with positive modifiers (e.g. "good", "great") and decrease the score of the article for sentences with negative modifiers (e.g. "bad" awful") and references to bad events (e.g. "shooting", "murder"). We then sum the scores across all of the articles and return the sorted list of cities. Since the user entered the string "Best parks," we return the list sorted descending by score

  Remove the System Entirely

  Another option would be to remove the current UA in its entirety. The current UA is complicated and takes a long time to run. Unstructured analytics could still be performed, but not while the application is running live. We alternatively could perform UA, searching through Yelp and other sources, to find the number of high-quality parks, restaurants, and other locations in and around each city. In this case, high-quality might mean "with a rating greater than 4 stars," for example. We could then store those results as structured data and perform a TOPSIS analysis on them to sort the user's final list of cities based on the user's preferences.

- **Restrict Unstructured Analysis(UA) Input:**

  The idea behind the UA section of the application was to allow users to input anything we might not have considered while creating the application. However, in its current state where the user can enter literally any string, it allows too much variety to be reasonable. A different approach would be to allow a modifier - e.g. "Best" - and a single object it modifies - e.g. "coffee", "hotels", "parks", etc. That way, the application only has to deal with a set of queries that takes the form of "Best coffee" or "Worst hotels," which is far more manageable.

- **Change the Problem Statement:**

  The application, as it stands, is somewhat limited by the problem statement itself. Finding a sister city places a greater emphasis on the input city rather than the user's preferences. An alternative, more user-focused problem might be to find the user's ideal city to live in. For such a problem, the application might ask questions about the user's preferences. Does the user like the outdoors? Does the user like to travel? What kind of job would the user like to have? The application could then take those answers and use them to adjust weights for analysis. A user that likes the outdoors might be suggested cities with a lot of national parks nearby. A user that wants a job in the tech industry might be suggested cities with lots of technological development. Much of the same data for the current problem could also be used for this problem.

- **Creation of a more Robust Validation Dataset/metric:**

  As of now we validate our unstructured analysis with embeddings visualization,but we can have a more robust validation dataset or metrics to validate our unstructured analysis results for ranking th relevance of query to the articles analyzed for a particular city.

7. CONCLUSIONS

The finding sister city project is an application which is our interpretation of what sister city should mean. Sister city means cities which are similar in user-defined criteria. The criteria include Proximity to the current city, Demographics, Economy, Politics and Climate. Using the KNN technique that utilizes the city data along with the weighting categories, the application returns a list of 5 cities which are most relevant. User then enters a refinement term which is used for the unstructured analysis to return a ranked list. This analysis uses a trained Word2Vec model.

The application is limited in certain areas such as the scope of cities and data, the method of obtaining news articles,the effectiveness of unstructured analysis, and the way Sister city is defined.

Many of the limitations could be overcome in the future implementations of the application. By increasing the sources of data and number of cities. Taking a different approach to the Unstructured Analysis by building a rule-based system to more appropriately understand the user's refinement term and return results more contextually.

Below is the repository of the project: `https://github.ncsu.edu/rwblack/csc495-dddm`

8. REFERENCES

[1] *Sister Cities Policy (2017)* Retrieved from: `https://www.livingstone.qld.gov.au/DocumentCenter/Home/View/12442`

[2] B. Mitra, E. Nalisnick, N. Craswell, R. Caruana *Dual Embedding Space-Model proposed in research paper âĂIJA Dual Embedding Space Model for Document Ranking* Retrieved from: `https://arxiv.org/abs/1602.01137`