天津大學

本科生实训报告

实训单位	Pole Universitaire Leonard de Vinci, France
姓　　名	周菁涛
学　　号	3016218162
专　　业	软件工程
年　　级	2016 级
班　　级	四班

智能与计算学部软件学院

2019 年 6 月

CONTENT

CHAPTER I - TEAM INTRODUCTION

1.1 Team Members Introduction

1.1.1 Overall Introduction

The team consists of four members, Jingtao Zhou, Huanyu Zhang, Huimin Wang and Chunyang Liu, respectively. Each of our four teammates has his specialized field and contributed to our project in his own way. The workload to each person is properly balanced.

1.1.2 Member Preference & Specialized Areas

The working preference and specialized field for each of our team members is demonstrated as below:

Jingtao Zhou

- GUI design / User experience design
- Theoretical research on machine learning models
- Frontend programming
- Project management & leadership
- Timeline & group meeting scheduling
- Presentation design

Huanyu Zhang

- Database programming
- Backend API programming
- Architecture & Software engineering practices

Chunyang Liu

- Database programming
- Backend API programming
- Architecture & Software engineering practices

Huimin Wang

- Linux operation & maintenance
- Visualization
- CI & application deployment

1.2 Detailed Work Repartition

The project consists of mainly five sections: the initialization step, the elaboration step, the construction step, the deployment step, and the timeline of work repartition. The construction step is divided into three subsections, demonstrating the work of machine learning, visualization, and application development, respectively. For each section of our projecct, the detailed work that each student does is presented as below:

1.2.1 Initialization Step

Tasks

1. Study the problem context by choosing the data you want to mine.
2. Elaborate the Use-case diagram and detailed description of the most important cases.
3. Define the global architecture of the Project.

Timeline & Work Repartition

Table 1-1. Scheduling Details.

Task ID	Collaborators	Estimated Schedule
I-1	All members	Apr 29 - May 01
I-2	Chunyang, Huanyu, Huimin	Apr 29 - May 06
I-3	Jingtao	May 01 - May 03

1.2.2 Elaboration step

Given Tasks

1. Detailed architecture of the Project by describing all the functionalities and the employed languages.
2. Scraping and collect the data.
3. Data cleaning and transformation.
4. Analysis of the dataset.
   - a. Analysis of the GeoNames dataset.
   - b. Analysis of the GeoNames `JOIN` GapMinder Countries dataset.

Additional Tasks

5. Graphical user interface (GUI) prototyping & design.
6. API documents drafting.
7. GUI Frontend implementation (Static).

Timeline & Work Repartition

Table 1-2. Scheduling Details.

Task ID	Collaborators	Estimated Schedule
II-1	Huanyu, Chunyang	May 06 - May 07
II-2	All members	May 07
II-3	All members	May 07 - May 09
II-4a	Jingtao	May 09 - May 14
II-4b	Huanyu, Chunyang, Huimin	May 09 - May 14
II-5	Jingtao	May 06 - May 10
II-6	Huanyu, Chunyang	May 10 - May 14
II-7	Jingtao	May 10 - May 15

1.2.3 Construction step

Given Tasks

1. Integration of all the cases defined in the elaboration step.
2. Machine Learning.
   - a. Clustering. Candidate models: K-Means, DBScan, SpectralClustering.
   - b. Classification Model Training. Model: Recurrent Neural Network.
   - c. Regression Model Training. Which model to use is still under research, but likely to be Recurrent Neural Network as well. We're still dealing with potential issues brought by the non-Cartesian output values (global coordinates).
3. Visualization of the dataset.
4. Program the application and make the main tests.
   - a. Backend programming - Database & API Implementation.
   - b. Frontend-backend docking.
   - c. GUI Frontend completion (Interactive).

Additional Tasks

5. Model & parameters optimization (If applicable).

Timeline & Work Repartition

Table 1-3. Scheduling Details.

Task ID	Collaborators	Estimated Schedule
III-1	-	-
III-2a	Jingtao, Huanyu	May 13 - May 19

Task ID	Collaborators	Estimated Schedule
III-2b	Jingtao, Huanyu	May 13 - May 19
III-2c	All members	May 19 - May 26
III-3	Jingtao	May 15 - May 20
III-4a	Huanyu, Chunyang	May 20 - May 24
III-4b	All members	May 24 - May 27
III-4c	Jingtao	May 24 - May 27
III-5	All members	May 20 - May 25

1.2.4 Deployment and Reporting

Given Tasks

1. Deploy the project if possible in the defined environment
2. Prepare a detailed report
3. Presentation

Additional Tasks

4. Continuous integration (CI) deployment

Timeline & Work Repartition

Table 1-4. Scheduling Details.

Task ID	Collaborators	Estimated Schedule
IIII-1	Huimin	May 27 - May 29
IIII-2	All members	May 29 - June 14
IIII-3	All members	May 29 - June 05
IIII-4	Huimin	May 29 - June 04

CHAPTER II - SYSTEM REQUIREMENT ANALYSIS

2.1 Background

In the past few decades, the contribution of computers to social productivity mainly lies in its ability to process structural data, such as numerical values, forms, and single colors. People can convert real-life data into structural data and store it in a computer. This liberates a large number of the simplest repetitive workforce.

In recent years, *neural networks* have increased the ability of computers to process non-structural data, or semi-structural data, such as music, images, and text paragraphs; This made abstracting and extracting features of large-scale data possible.

A very important application of machine learning and neural networks is Natural Language Processing. Unlike programming languages, natural language is often more complex, bulky, and difficult to logically describe. The IT industry has introduced abundant research and solutions in this field. When we talk about natural language processing, however, it is usually about a semantic level of processing. **In addition to this, there is an area that could be easily overlooked - the abstraction of the *spelling features* of different writing systems.**

Figure 2-1. Placenames in Vietnam

When searching for available datasets related to language and geographic information, we found an open database provided by GeoNames that provides information of the names, locations, countries, and attributes of a large number of geographic features covering the earth. In this experiment, we used data sets provided by GeoNames, combined with `PyTorch`, `Matplotlib` and other data processing and machine learning tools to study the connection between geographic location and city name.

2.2 Problem Description

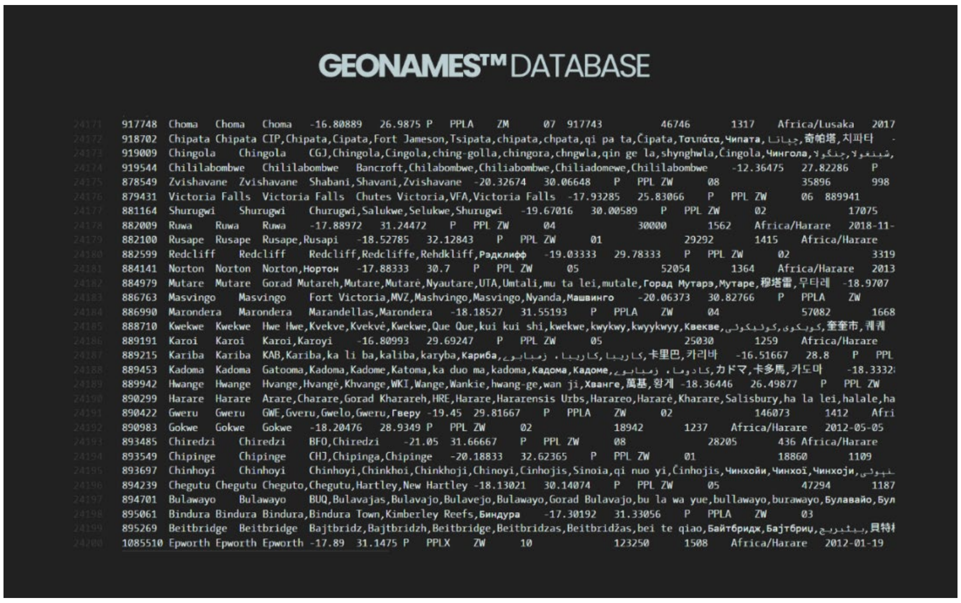
The goal of the project is to abstract the relationship between the spelling of the place name and its geographical location from the GeoNames and GapMinder dataset, and train a recurrent neural network (RNN) on certain goals. **The implemented application should demonstrate the insights, statistics, and facts that we discovered, and predict the approximate region of the place name by its spelling.** The regions are clustered with the help of GapMinder countries data.

In the later stages of the project, we will explore and summarize the relationship between place name spellings and their coordinates revealed in the training results.

CHAPTER III - SYSTEM DESIGN OVERVIEW

3.1 Dataset Overview

3.1.1 GeoNames Database



```
917748	Choma	Choma	Choma	-16.80889	26.9875	P	PPLA	ZM		07	917743			46746		1317	Africa/Lusaka	2017
918702	Chipata	Chipata	CIP,Chipata,Cipata,Fort Jameson,Tsipata,chipata,chpata,qi pa ta,Čipata,Τσιπάτα,Чипата,چیپاتا,奇帕塔,치파타	-
919009	Chingola	Chingola	CGJ,Chingola,Cingola,ching-golla,chingora,chngwla,qin ge la,shynghwla,Čingola,Чингола,چنگولا,شينغولا,
919544	Chililabombwe	Chililabombwe	Bancroft,Chilabombwe,Chiliabombwe,Chiliadomewe,Chililabombwe	-12.36475	27.82286	P
878549	Zvishavane	Zvishavane	Shabani,Shavani,Zvishavane	-20.32674	30.06648	P	PPL	ZW		08				35896		998
879431	Victoria Falls	Victoria Falls	Chutes Victoria,VFA,Victoria Falls	-17.93285	25.83066	P	PPL	ZW		06	889941
881164	Shurugwi	Shurugwi	Churugwi,Salukwe,Selukwe,Shurugwi	-19.67016	30.00589	P	PPL	ZW		02				17075
882009	Ruwa	Ruwa	Ruwa	-17.88972	31.24472	P	PPL	ZW		04				30000		1562	Africa/Harare	2018-11-
882100	Rusape	Rusape	Rusape,Rusapi	-18.52785	32.12843	P	PPL	ZW		01				29292		1415	Africa/Harare
882599	Redcliff	Redcliff	Redcliff,Redcliffe,Rehdkliff,Рэдклифф	-19.03333	29.78333	P	PPL	ZW		02				3319
884141	Norton	Norton	Norton,Нортон	-17.88333	30.7	P	PPL	ZW		05				52054		1364	Africa/Harare	2013
884979	Mutare	Mutare	Gorad Mutareh,Mutare,Mutarė,Nyautare,UTA,Umtali,mu ta lei,mutale,Горад Мутарэ,Мутаре,穆塔雷,무타레	-18.9707
886763	Masvingo	Masvingo	Fort Victoria,MVZ,Mashvingo,Masvingo,Nyanda,Машвинго	-20.06373	30.82766	P	PPLA	ZW
886990	Marondera	Marondera	Marandellas,Marondera	-18.18527	31.55193	P	PPLA	ZW		04				57082		1668
888710	Kwekwe	Kwekwe	Hwe Hwe,Kvekve,Kvekvė,Kwekwe,Que Que,kui kui shi,kwekwe,kwykwy,kwyykwyy,Квекве,کوئیکوئی,کویکوی,奎奎市,퀘퀘
889191	Karoi	Karoi	Karoi,Karoyi	-16.80993	29.69247	P	PPL	ZW		05				25030		1259	Africa/Harare
889215	Kariba	Kariba	KAB,Kariba,ka li ba,kaliba,karyba,Кариба,زمبابوے . کاریبا,کاریبا,卡里巴,카리바	-16.51667	28.8	P	PPL
889453	Kadoma	Kadoma	Gatooma,Kadoma,Kadome,Katoma,ka duo ma,kadoma,Кадома,Кадоме,زمبابوے . کادوما,カドマ,卡多馬,카도마	-18.3332
889942	Hwange	Hwange	Hvange,Hvangė,Khvange,WKI,Wange,Wankie,hwang-ge,wan ji,Хванге,萬基,황게	-18.36446	26.49877	P	PPL	ZW
890299	Harare	Harare	Arare,Charare,Gorad Kharareh,HRE,Harare,Hararensis Urbs,Harareo,Hararė,Kharare,Salisbury,ha la lei,halale,ha
890422	Gweru	Gweru	GWE,Gveru,Gwelo,Gweru,Гверу	-19.45	29.81667	P	PPLA	ZW		02				146073		1412	Afri
890983	Gokwe	Gokwe	Gokwe	-18.20476	28.9349	P	PPL	ZW		02				18942		1237	Africa/Harare	2012-05-05
893485	Chiredzi	Chiredzi	BFO,Chiredzi	-21.05	31.66667	P	PPL	ZW		08				28205		436	Africa/Harare
893549	Chipinge	Chipinge	CHJ,Chipinga,Chipinge	-20.18833	32.62365	P	PPL	ZW		01				18860		1109
893697	Chinhoyi	Chinhoyi	Chinhoyi,Chinkhoi,Chinkhoji,Chinoyi,Cinhojis,Sinoia,qi nuo yi,Činhojis,Чинхойи,Чинхої,Чинхоји,
894239	Chegutu	Chegutu	Cheguto,Chegutu,Hartley,New Hartley	-18.13021	30.14074	P	PPL	ZW		05				47294		1187
894701	Bulawayo	Bulawayo	BUQ,Bulavajas,Bulavajo,Bulavejo,Bulawayo,Gorad Bulavajo,bu la wa yue,bullawayo,burawayo,Булавайо,Бул
895061	Bindura	Bindura	Bindura,Bindura Town,Kimberley Reefs,Биндура	-17.30192	31.33056	P	PPLA	ZW		03
895269	Beitbridge	Beitbridge	Bajtbridz,Bajtbridzh,Beitbridge,Beitbridzas,Beitbridžas,bei te qiao,Байтбридж,Бајтбриџ,بیتبریج,貝特
1085510	Epworth	Epworth	Epworth	-17.89	31.1475	P	PPLX	ZW		10				123250		1508	Africa/Harare	2012-01-19
```

Figure 3-1(a). GeoNames Database Overview.

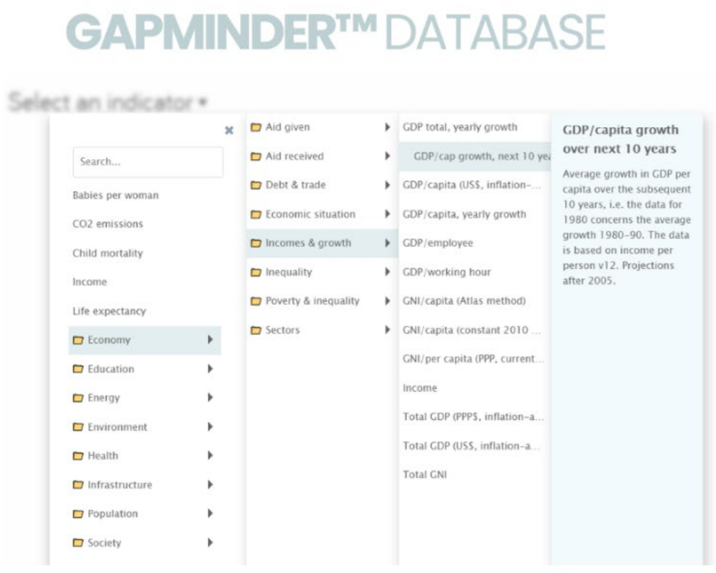
```
geonameid         integer id of record in geonames database
name              name of geographical point (utf8) varchar(200)
asciiname         name of geographical point in plain ascii characters; varchar(200)
alternatenames    alternatenames; comma separated; ascii names automatically
transliterated; convenience attribute from alternatename table; varchar(10000)
latitude          latitude in decimal degrees (wgs84)
longitude         longitude in decimal degrees (wgs84)
feature class     see http
feature code      see http
country code      ISO-3166 2-letter country code; 2 characters
cc2               alternate country codes; comma separated; ISO-3166 2-letter country
code; 200 characters
admin1 code       fipscode (subject to change to iso code); see exceptions below; see
file admin1Codes.txt for display names of this code; varchar(20)
admin2 code       code for the second administrative division; a county in the US; see
file admin2Codes.txt; varchar(80)
admin3 code       code for third level administrative division; varchar(20)
admin4 code       code for fourth level administrative division; varchar(20)
population        bigint (8 byte int)
elevation         in meters; integer
dem               digital elevation model; srtm3 or gtopo30; average elevation of
3''x3'' (ca 90mx90m) or 30''x30'' (ca 900mx900m) area in meters; integer. srtm processed
by cgiar/ciat.
timezone          the iana timezone id (see file timeZone.txt) varchar(40)
modification date date of last modification in yyyy-MM-dd format
```

Figure 3-1(b). GeoNames Database Overview.

The GeoNames geographical database contains over 10 million geographical names and consists of over 9 million unique features with 2.8 million populated places and 5.5 million alternate names.

All features are categorized into one out of nine feature classes and further subcategorized into one out of 645 feature codes.

3.1.2 Gapminder Countries Data



GDP/capita growth over next 10 years

Average growth in GDP per capita over the subsequent 10 years, i.e. the data for 1980 concerns the average growth 1980-90. The data is based on income per person v12. Projections after 2005.

Figure 3-2. GapMinder Database Overview.

Gapminder is an independent Swedish foundation with no political, religious or economic affiliations. Gapminder is a fact tank, not a think tank. Gapminder fights devastating misconceptions about global development.

Gapminder produces free teaching resources making the world understandable based on reliable statistics. Gapminder promotes a fact-based worldview everyone can understand. Gapminder collaborates with universities, UN, public agencies and non-governmental organizations. All Gapminder activities are governed by the board. We do not award grants.

Abundant data of countries all over the world could be found on the Gapminder Website.

3.2 Global Architecture

Considering that the main machine learning task contains only a single input and output, the project is not too complicated at the architectural level. We are suggested to put a part of data visualization results and analysis in the application interface also. This project can be split into the following modules from the perspective of global architecture.

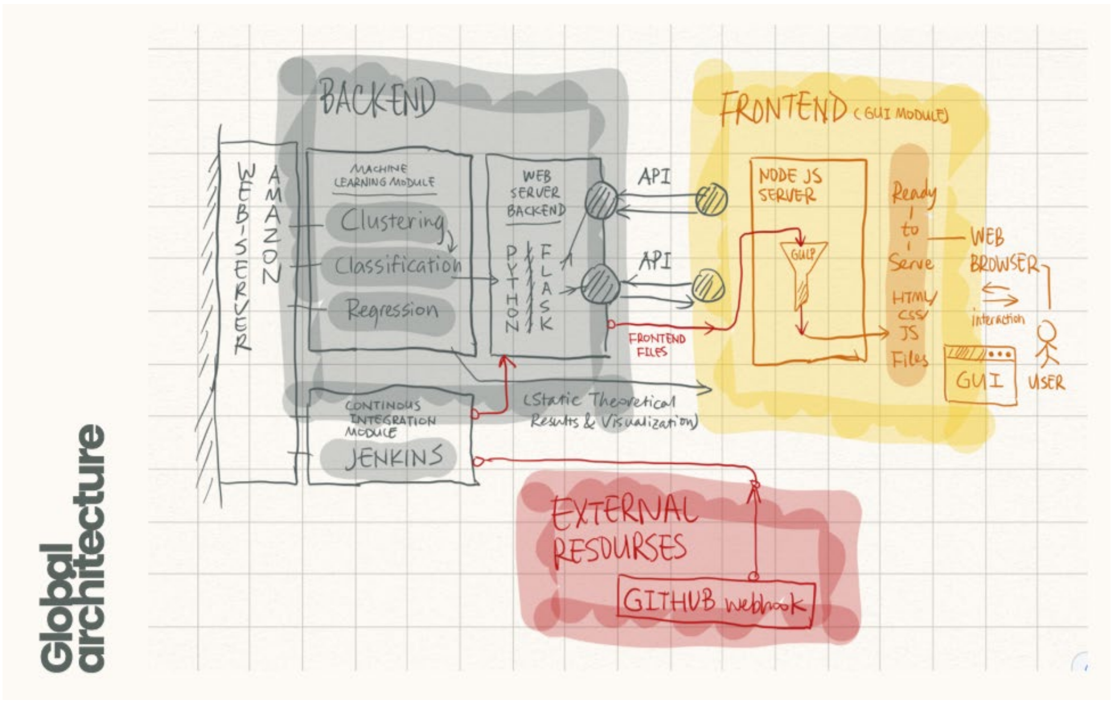


Figure 3-3. Global Architecture.

As seen from the image above, all modules in this project are deployed on the same Amazon web server but properly decoupled.

3.2.1 GUI Module

The frontend uses the Web as the application interface because the Web has become the only *de-facto* cross-platform, universal interface standard in the IT industry. We use HTML to complete the markup documentation, SASS to write styles, JavaScript to implement web requests and interaction logic, and to automate development through Gulp-like front-end modern workflow tools. The source code is compiled into a static web file via Gulp, and served afterward.

3.2.2 Machine Learning Module

This section contains clustering, classification, and regression, where the output of the cluster is input to the classification. Regression is used only for model comparison in theoretical research and does not have a significant impact on the function of the application. Most of the code for this module is written using Python and related machine learning libraries. It is worth noting that the neural network after machine learning training is stored as a cache on the server, rather than training a new model separately each time the user requests it. We believe this helps maintain performance consistency of our application and saves computing resources.

3.2.3 API Module

This module is responsible for the API of the program, which works as a bridge communicating the results of machine learning and the input/output of the graphical user interface. We use Flask to implement this backend part of the web development.

3.2.4 Continuous Integration Module

Deploying continuous integration helps automate the entire software development process, eliminating the need for manual testing, compilation, and deployment for each update. We use Jenkins to complete the CI module and capture real-time code updates uploaded to GitHub via tools including Webhook.

3.3 UML Diagrams

3.3.1 Use-case Diagrams

For this part, we designed a use case diagram to show the relationship between our various use cases, such as the relationship between the backend and the user, the relationship between the front end and the user, and some interaction between them.

3.3.2 Class Diagrams

Immediately after we created the class diagram to represent the relationship between our various classes, we designed six classes, namely: Doa, MachineLearning, Statistics, Browser, Classfication, Regression. Through these six classes to expand our specific jobs.

3.3.3 Sequential Diagrams

The Sequential diagram is a diagram that shows the relationship between our specific projects. It describes the complete flow and interaction details of our entire program by describing the operations between the front end, the back end, and the user.

We have designed two Sequential diagrams to represent our two initial ideas for the overall architecture of the program. Finally, we will start our work based on these two Sequential diagrams.

CHAPTER IV - MODEL RESEARCH & THEORETICAL DETAILS

4.1 PCA (Principal Component Analysis)

4.1.1 Overview

Principal component analysis (PCA) is one of the most widely used data dimensionality reduction algorithms. The main idea of PCA is to map n-dimensional features to k-dimension, which is a new orthogonal feature, also called principal component, which is a k-dimensional feature reconstructed from the original n-dimensional features.

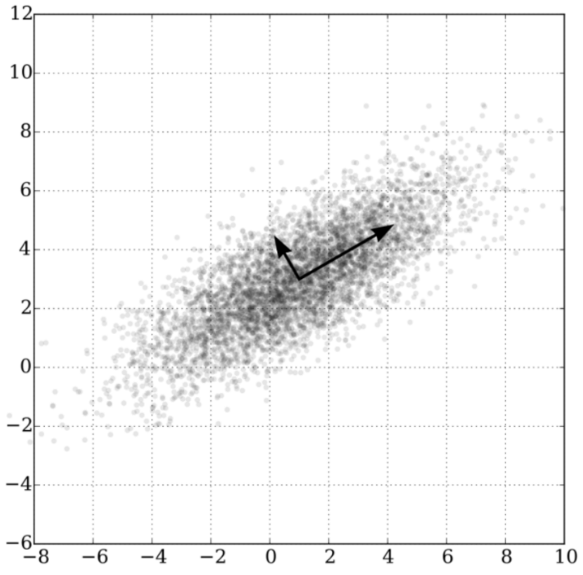


Figure 4-1. PCA method visualized.

4.1.2 Mathematical Description

Define a matrix of n × m, XT is the data of the de-average (moving to the origin centered on the average), and the behavior data samples are listed as data categories (note that XT is defined here instead of X). Then the singular value of X is decomposed into X = W Σ VT, where m × m matrix W is the eigenvector (eigenvector) matrix of XXT, Σ is a non-negative rectangular diagonal matrix of m × n, and V is n × n of XTX Eigenvector (feature vector) matrix. According to this,

$$\begin{aligned}\boldsymbol{Y}^\top &= \boldsymbol{X}^\top \boldsymbol{W} \\ &= \boldsymbol{V}\boldsymbol{\Sigma}^\top \boldsymbol{W}^\top \boldsymbol{W} \\ &= \boldsymbol{V}\boldsymbol{\Sigma}^\top\end{aligned}$$

When m < n − 1, V is not uniquely defined under normal circumstances, and Y is uniquely defined. W is an orthogonal matrix, YTWT=XT, and the first column of YT consists of the first principal component, the second column consists of the second principal component, and so on.

To get an effective way to reduce the data dimension, we can use WL to map X to a low-dimensional space that only applies the first L vectors:

$$\mathbf{Y} = \mathbf{W_L}^\top \mathbf{X} = \mathbf{\Sigma_L} \mathbf{V}^\top$$

The single vector matrix of X is equivalent to the eigenvector of the covariance matrix C = X XT,

$$\mathbf{X}\mathbf{X}^\top = \mathbf{W}\mathbf{\Sigma}\mathbf{\Sigma}^\top \mathbf{W}^\top$$

Given a set of points in the Euclidean space, the first principal component corresponds to a line passing through the multidimensional space averaging points while ensuring that the sum of the squares of the distances from each point to the straight line is minimal. After the first principal component is removed, the second principal component is obtained in the same manner. So on and so forth.

4.1.3 Applications in This Project

In this project, PCA is used to deal with the dimensionality reduction of pre-cluster data preprocessing.

Due to the need to cluster different countries or regions, and describe the indicators, the size, population, language, economic level, income level, geographical location and their respective growth of each country entity, etc. There is a lot of data that is highly correlated (such as the growth rate of income levels and the growth rate of economic levels). Therefore, you need to call PCA for dimensionality reduction to optimize subsequent computing performance.

4.2 CLUSTERING ALGORITHMS

4.2.1 Overview

The K-Means algorithm is an unsupervised clustering algorithm, which is simple to implement and has a good clustering effect, so it is widely used. For a given sample set, the algorithm divides the sample set into K clusters according to the distance between the samples. The process of describing the execution in a popular way is to make the points in the cluster as close together as possible, and to make the distance between the clusters as large as possible.

4.2.2 Programmatic Implementation

```
def generate_k(data_set, k):
    """
   Given `data_set`, which is an array of arrays,
   find the minimum and maximum for each coordinate, a range.
   Generate `k` random points between the ranges.
   Return an array of the random points within the ranges.
   """
    centers = []
    dimensions = len(data_set[0])
    min_max = defaultdict(int)

    for point in data_set:
        for i in xrange(dimensions):
            val = point[i]
            min_key = 'min_%d' % i
            max_key = 'max_%d' % i
            if min_key not in min_max or val < min_max[min_key]:
                min_max[min_key] = val
            if max_key not in min_max or val > min_max[max_key]:
                min_max[max_key] = val

    for _k in xrange(k):
        rand_point = []
        for i in xrange(dimensions):
            min_val = min_max['min_%d' % i]
            max_val = min_max['max_%d' % i]

            rand_point.append(uniform(min_val, max_val))

        centers.append(rand_point)

    return centers


def k_means(dataset, k):
    k_points = generate_k(dataset, k)
    assignments = assign_points(dataset, k_points)
    old_assignments = None
    while assignments != old_assignments:
        new_centers = update_centers(dataset, assignments)
        old_assignments = assignments
        assignments = assign_points(dataset, new_centers)
    return zip(assignments, dataset)
```

4.2.3 Comparison to Alternative Models

This example shows characteristics of different clustering algorithms on datasets that are “interesting” but still in 2D. With the exception of the last dataset, the parameters of each of these dataset-algorithm pairs has been tuned to produce good clustering results. Some algorithms are more sensitive to parameter values than others.

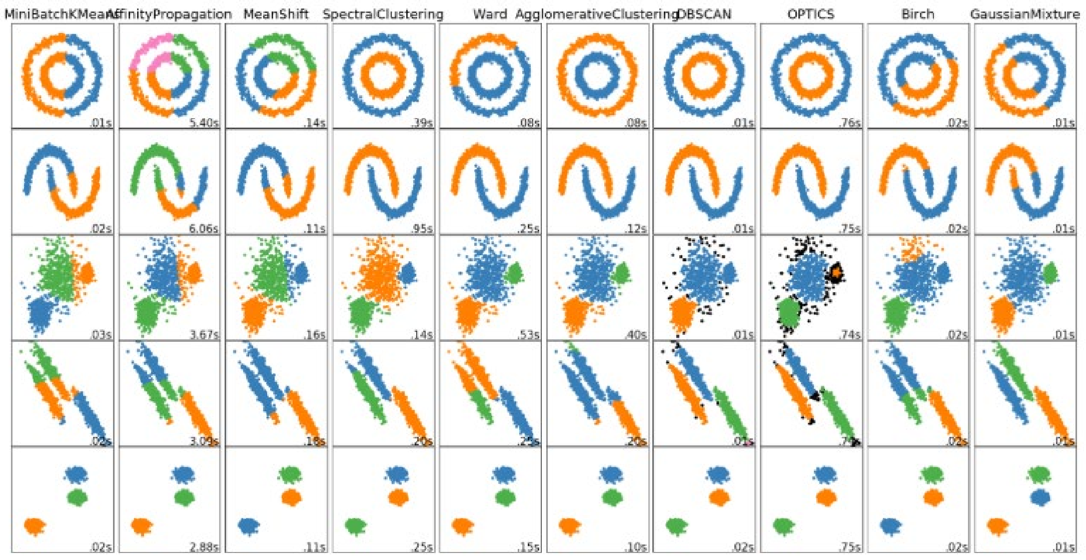


Figure 4-2. Comparison between different clustering models.

The last dataset is an example of a 'null' situation for clustering: the data is homogeneous, and there is no good clustering. For this example, the null dataset uses the same parameters as the dataset in the row above it, which represents a mismatch in the parameter values and the data structure.

While these examples give some intuition about the algorithms, this intuition might not apply to very high dimensional data.

4.2.4 Applications in This Project

The main process of this experiment can be abstracted into a classification problem, and the number of classification tags greatly affects the experimental results. The initial data that can be obtained is the name of each city and the country it belongs to, and there are more than 240 total countries, which is inconvenient for classification.

We have thought about replacing countries/regions with continents, but there are also two significant problems. One is that the number of continents is too small, and the other is that continents can only reflect geographical differences and do not reflect the language of the country. Differences between culture and urban naming habits (for example, Australia and North America score two continents, but naming conventions and language similarities are large). Therefore, a separate process is needed to perform K-means clustering for each country and region.

4.3 RNN (Recurrent Neural Network)

4.3.1 Overview

For neural networks, the most basic model is the ordinary feedforward neural network. **There are two significant drawbacks to the feedforward neural network in processing and timing related data:**

- Each time the output of the network depends only on the current input, without considering the interaction of the inputs at different times;
- The dimensions of the input and output are fixed, without taking into account the insufficiency of the length of the sequence structure data.

Recurrent Neural Network (RNN) solves the above problem. It is a kind of neural network dedicated to processing time series data samples. Each layer of its output is not only output to the next layer, but also outputs a hidden state to the current The layer is used when processing the next sample. Just as convolutional neural networks can easily be extended to images with large widths and heights, and some convolutional neural networks can handle images of different sizes, and circular neural networks can be extended to longer sequence data, and most The cyclic neural network can process data of different sequence lengths (for loops, variable lengths). It can be seen as a fully connected neural network with self-loop feedback.

4.3.2 Model Visualization

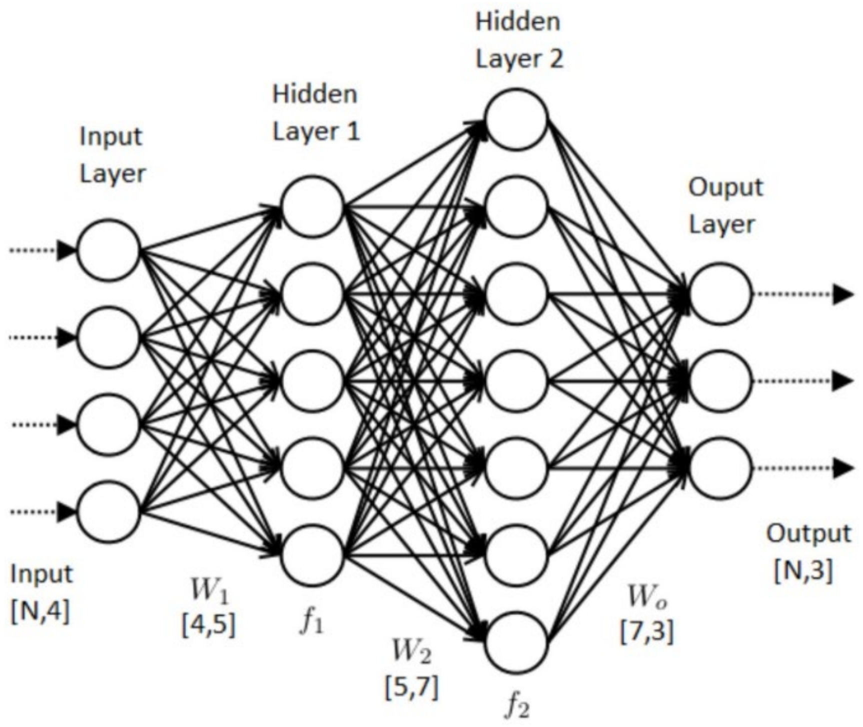


Figure 4-3(a). A Simple feedforward Networks & Recurrent Neural Network visualized.

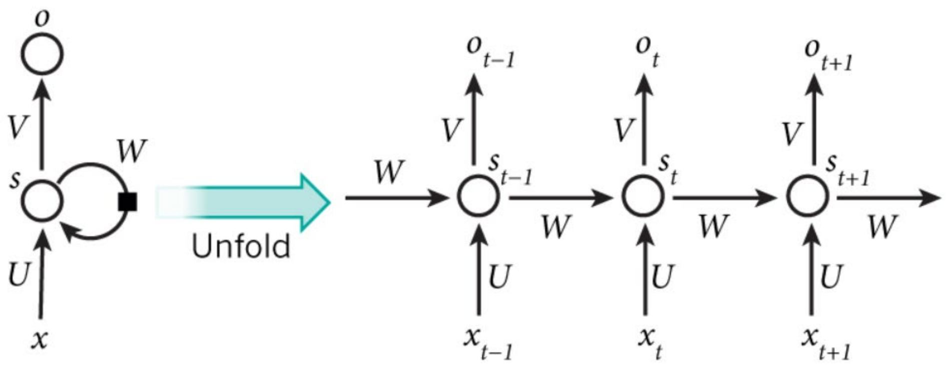
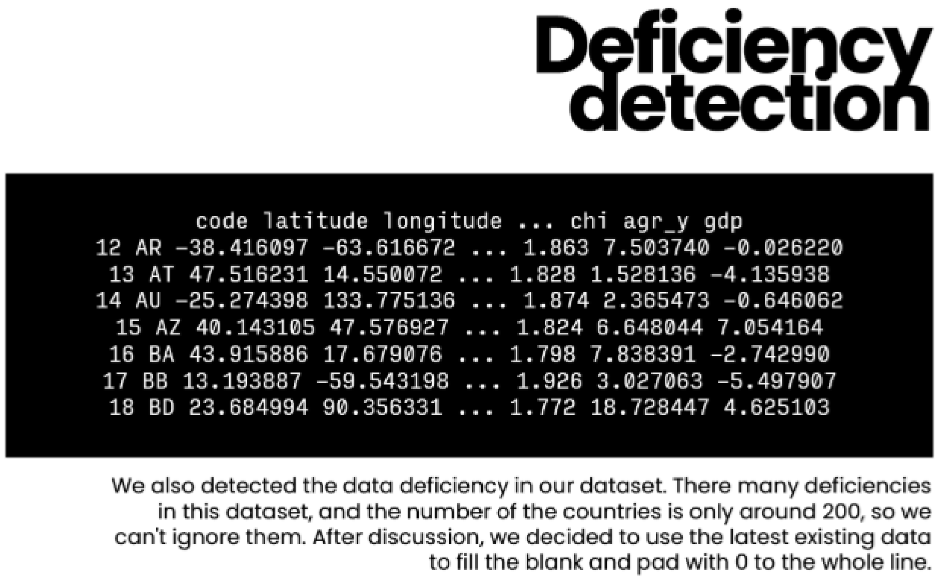


Figure 4-3(b). A Simple feedforward Networks & Recurrent Neural Network visualized.

4.3.3 Applications in This Project

The main input data of the model trained in this project is a word string with certain sequence characteristics, so it is suitable for description by RNN. In the actual algorithm, we need to convert the string into a tensor (Tensor) to input into the RNN for training. The specific conversion algorithm is described in the following section.

5.1 Data Scraping and Cleaning



Deficiency detection

```
     code latitude longitude ... chi agr_y gdp
  12 AR -38.416097 -63.616672 ... 1.863 7.503740 -0.026220
  13 AT 47.516231 14.550072 ... 1.828 1.528136 -4.135938
  14 AU -25.274398 133.775136 ... 1.874 2.365473 -0.646062
   15 AZ 40.143105 47.576927 ... 1.824 6.648044 7.054164
  16 BA 43.915886 17.679076 ... 1.798 7.838391 -2.742990
  17 BB 13.193887 -59.543198 ... 1.926 3.027063 -5.497907
  18 BD 23.684994 90.356331 ... 1.772 18.728447 4.625103
```

We also detected the data deficiency in our dataset. There many deficiencies in this dataset, and the number of the countries is only around 200, so we can't ignore them. After discussion, we decided to use the latest existing data to fill the blank and pad with 0 to the whole line.

Figure 5-1. Data deficiency detection.

This necessary section is implemented mainly by other team members in this project. In this section, we used Python to process, collect, clean and transform our data, making them as a valid input into our mathematical models.

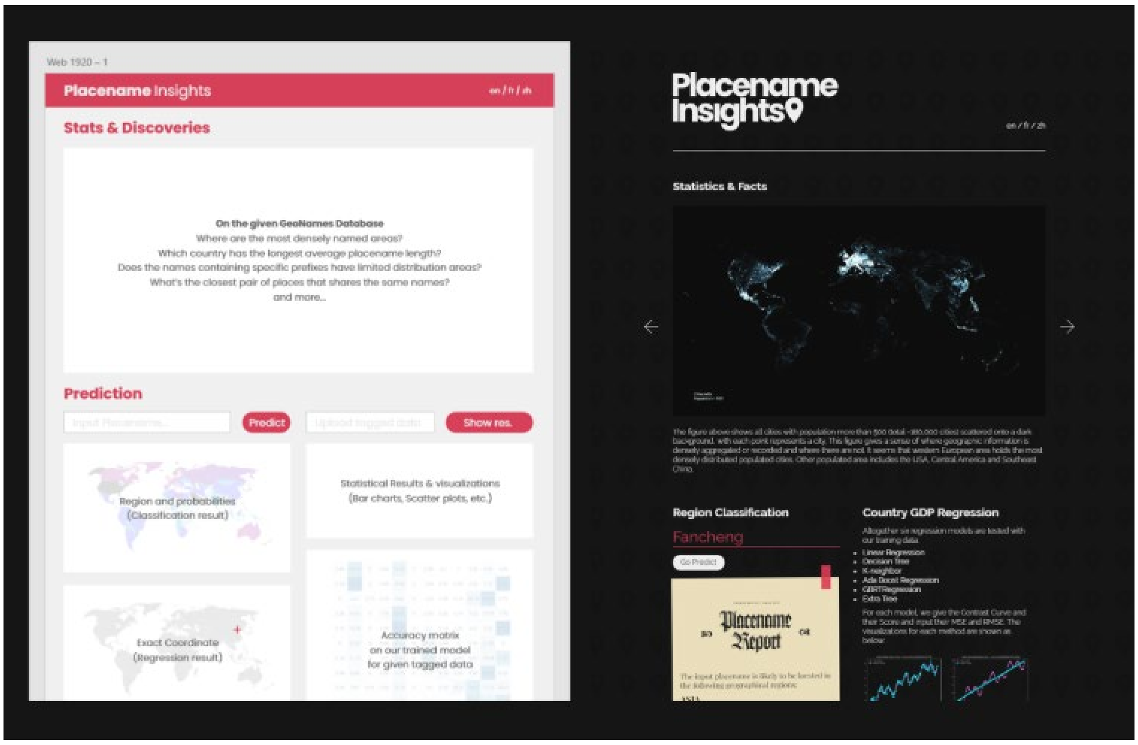
According to the rough check, only 0.1% of the cities lost their country codes, and almost every city has its `asciiname`. So we can ignore those lost their country codes or `asciiname` city to avoid the negative effect cased by them.

There many deficiencies in this dataset, and the number of the countries is only around 200, so we can't ignore them. After discussion, we decided to use the latest existing data to fill the blank and pad with `0` to the whole line.

A word to vector algorithm was implemented in order to input into our neural networks provided by PyTorch.

5.2 GUI Prototyping & Design

The graphical user interface, according to the global application structure discussed above, should contain mainly two parts: the machine learning input and output section, and the statistics, insights & visualization section. The interface should be a single-page web application (SPA). **Due to the initial design of our project, we do not need to store user data or input, test results. Instead, the interface responds at each anonymous request.**



On the given GeoNames Database
Where are the most densely named areas?
Which country has the longest average placename length?
Does the names containing specific prefixes have limited distribution areas?
What's the closest pair of places that shares the same names?
and more...

Region and probabilities (Classification result)

Statistical Results & visualizations (Bar charts, Scatter plots, etc.)

Exact Coordinate (Regression result)

Accuracy matrix on our trained model for given tagged data

The figure above shows all cities with population more than 500 (total ~180,000 cities) scattered onto a dark background, with each point represents a city. This figure gives a sense of where geographic information is densely aggregated or recorded and where there are not. It seems that western European area holds the most densely distributed populated cities. Other populated area includes the USA, Central America and Southeast China.

The input placename is likely to be located in the following geographical regions:
ASIA

Altogether six regression models are tested with our training data
- Linear Regression
- Decision Tree
- K-neighbor
- Ada Boost Regression
- GBRTRegression
- Extra Tree

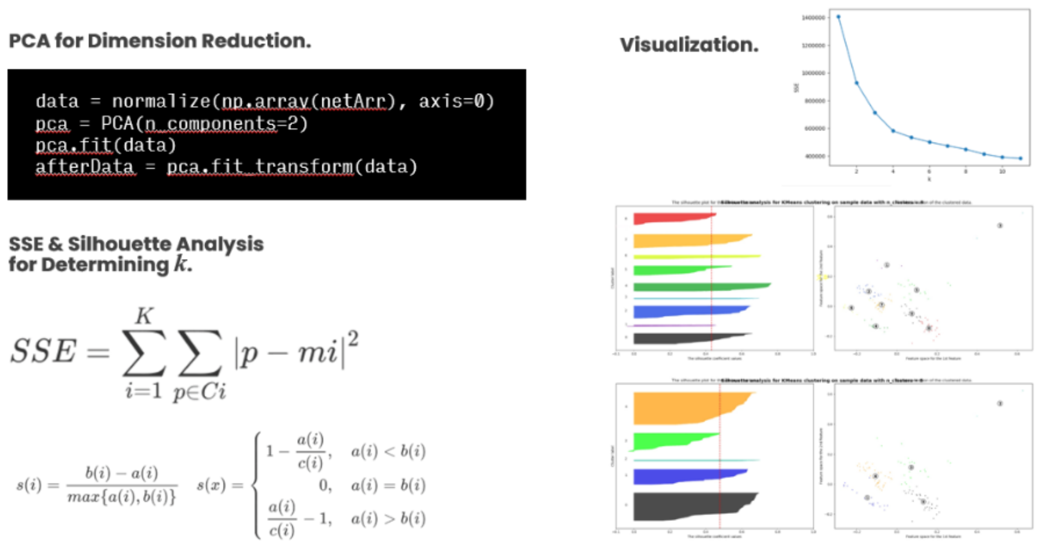
For each model, we give the Contrast Curve and their Score and input their MSE and RMSE. The visualizations for each method are shown as below.

Figure 5-2. Comparison of An early version of the GUI prototype & The implementation.

The visual design, on the other hand, focuses on typography, layout and visual consistency. Poppins font family is chosen for our identity font, and thus used across the website. We use different font weights to stress the contrast and layering of GUI, and the color magenta `#df3954` is chosen for theme color. **The entire interface is finally implemented on a dark background, which differs from the initial draft above, because the scatter plots look better this way.**

5.3 Machine Learning

5.3.1 Clustering



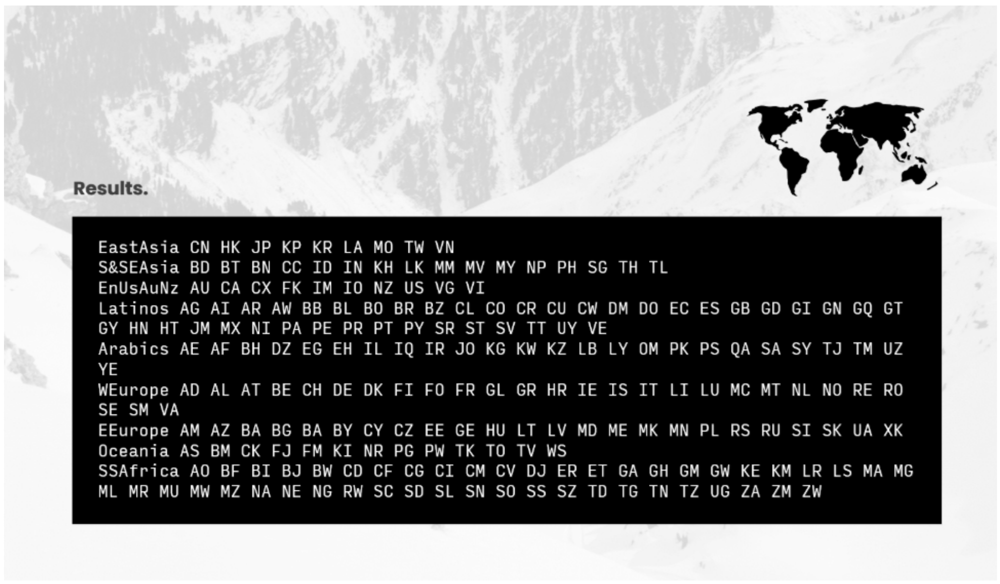
```
data = normalize(np.array(netArr), axis=0)
pca = PCA(n_components=2)
pca.fit(data)
afterData = pca.fit_transform(data)
```

$$SSE = \sum_{i=1}^{K} \sum_{p \in Ci} |p - mi|^2$$

$$s(i) = \frac{b(i) - a(i)}{max\{a(i), b(i)\}} \qquad s(x) = \begin{cases} 1 - \frac{a(i)}{c(i)}, & a(i) < b(i) \\ 0, & a(i) = b(i) \\ \frac{a(i)}{c(i)} - 1, & a(i) > b(i) \end{cases}$$

Figure 5-3. The clustering process summarized.

Before machine learning, we need to cluster this city by so we choose K-means model to do this job. Considering the vast difference between different countries, like location, culture, population, language, GDP and so on, so we use the Gapminder to do the K-means method, and we need to use the ***PCA*** method to do dimension reduction analysis.



```
EastAsia CN HK JP KP KR LA MO TW VN
S&SEAsia BD BT BN CC ID IN KH LK MM MV MY NP PH SG TH TL
EnUsAuNz AU CA CX FK IM IO NZ US VG VI
Latinos AG AI AR AW BB BL BO BR BZ CL CO CR CU CW DM DO EC ES GB GD GI GN GQ GT
GY HN HT JM MX NI PA PE PR PT PY SR ST SV TT UY VE
Arabics AE AF BH DZ EG EH IL IQ IR JO KG KW KZ LB LY OM PK PS QA SA SY TJ TM UZ
YE
WEurope AD AL AT BE CH DE DK FI FO FR GL GR HR IE IS IT LI LU MC MT NL NO RE RO
SE SM VA
EEurope AM AZ BA BG BA BY CY CZ EE GE HU LT LV MD ME MK MN PL RS RU SI SK UA XK
Oceania AS BM CK FJ FM KI NR PG PW TK TO TV WS
SSAfrica AO BF BI BJ BW CD CF CG CI CM CV DJ ER ET GA GH GM GW KE KM LR LS MA MG
ML MR MU MW MZ NA NE NG RW SC SD SL SN SO SS SZ TD TG TN TZ UG ZA ZM ZW
```

Figure 5-4. The clustering result.

```
data = normalize(np.array(netArr), axis=0)
pca = PCA(n_components=2)
pca.fit(data)
afterData = pca.fit_transform(data)
```

After dimension reduction, we need to decide the value of ***K***, we use ***sum of the squared errors*** and ***Silhouette analysis*** to decide the choose amount. Here are math theories of these two methods.

After clustering, to be compatible with *Geonames* and *Gapminder*, we use the ***country code*** to describe these countries.

5.3.2 Classification

Generate the RNN

RNN can be easily derived from the simple feed-forward neural network through PyTorch.

```
import torch.nn as nn

class RNN(nn.Module):
    def __init__(self, input_size, hidden_size, output_size):
        super(RNN, self).__init__()
        self.hidden_size = hidden_size
        self.i2h = nn.Linear(input_size + hidden_size, hidden_size)
        self.i2o = nn.Linear(input_size + hidden_size, output_size)
        self.softmax = nn.LogSoftmax(dim=1)

    def forward(self, input, hidden):
        combined = torch.cat((input, hidden), 1)
        hidden = self.i2h(combined)
        output = self.i2o(combined)
        output = self.softmax(output)
        return output, hidden

    def initHidden(self):
        return torch.zeros(1, self.hidden_size)

n_hidden = 128
rnn = RNN(n_letters, n_hidden, n_categories)
```

RNN is a variant of a neural network that still "keep a memory" to the content of the previous input sequence. It has similar input, output, and a hidden layers module. The network is visualized as follows:

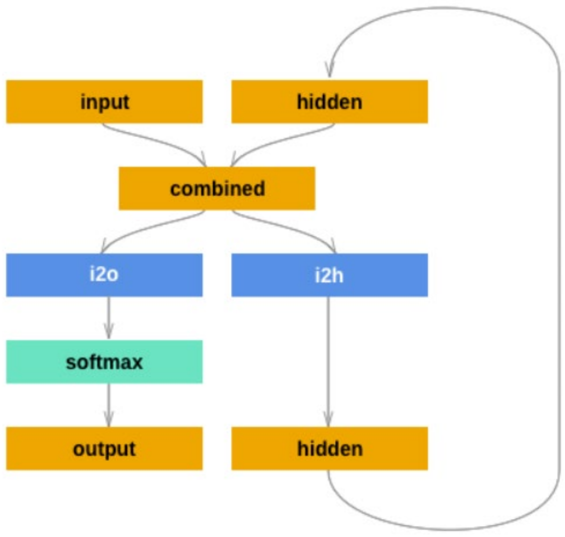


Figure 5-5. The applied RNN model explained.

Training the Network

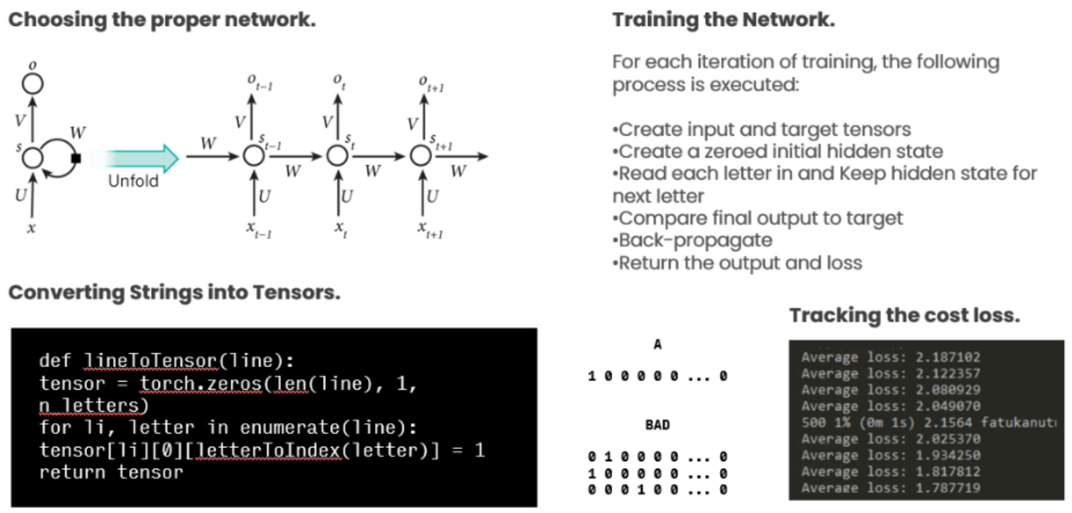
After defining several helper functions, we can begin the main process of training. The purpose of training as a whole is to reduce the cost function by guessing the results and comparing the correct results with feedback and constantly adjusting the parameters of each neuron in the neural network. In this project, `train.py` is the pipeline of the training process, where the `train` function declares a single training process:

```
learning_rate = 0.003

def train(category_tensor, line_tensor):
    outputt = 0
    loss = 0
    hidden = rnn.initHidden()

    rnn.zero_grad()

    for i in range(line_tensor.size()[0]):
        outputt, hidden = rnn(line_tensor[i], hidden)

    loss = criterion(outputt, category_tensor)
    loss.backward()

    for p in rnn.parameters():
        p.data.add_(-learning_rate, p.grad.data)

    return outputt, loss.item()
```

For each iteration of training, the following process is executed:

- Create input and target tensors
- Create a zeroed initial hidden state

- Read each letter in and Keep hidden state for next letter
- Compare final output to target
- Back-propagate
- Return the output and loss


Choosing the proper network.

Training the Network.

For each iteration of training, the following process is executed:

•Create input and target tensors
•Create a zeroed initial hidden state
•Read each letter in and Keep hidden state for next letter
•Compare final output to target
•Back-propagate
•Return the output and loss

Converting Strings into Tensors.

```
def lineToTensor(line):
tensor = torch.zeros(len(line), 1,
n_letters)
for li, letter in enumerate(line):
tensor[li][0][letterToIndex(letter)] = 1
return tensor
```

```
        A

1 0 0 0 0 0 ... 0


       BAD

0 1 0 0 0 0 ... 0
1 0 0 0 0 0 ... 0
0 0 0 1 0 0 ... 0
```

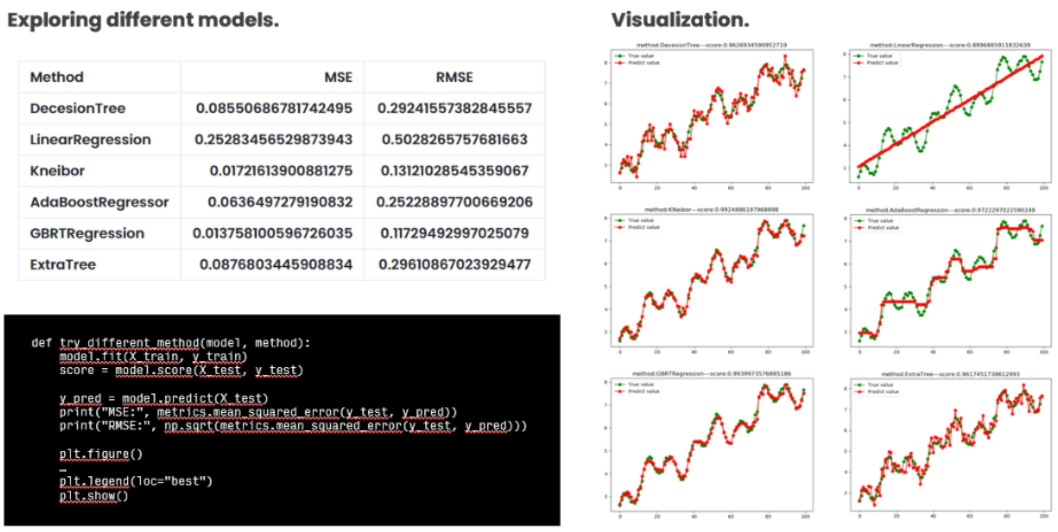
Tracking the cost loss.

```
Average loss: 2.187102
Average loss: 2.122357
Average loss: 2.080929
Average loss: 2.049070
500 1% (0m 1s) 2.1564 fatukanut
Average loss: 2.025370
Average loss: 1.934250
Average loss: 1.817812
Average loss: 1.787719
```

Figure 5-6. The classification process summarized.

5.3.3 Regression

This necessary section is implemented mainly by other team members in this project. They use the regression model to train our data. Although the regression is not suitable for our data for application development, we still need to carry out regression analysis and get key data such as MSE and RMSE.


Exploring different models.

Method	MSE	RMSE
DecesionTree	0.08550686781742495	0.29241557382845557
LinearRegression	0.25283456529873943	0.5028265757681663
Kneibor	0.01721613900881275	0.13121028545359067
AdaBoostRegressor	0.0636497279190832	0.25228897700669206
GBRTRegression	0.013758100596726035	0.11729492997025079
ExtraTree	0.0876803445908834	0.29610867023929477

```
def try_different_method(model, method):
    model.fit(X_train, y_train)
    score = model.score(X_test, y_test)

    y_pred = model.predict(X_test)
    print("MSE:", metrics.mean_squared_error(y_test, y_pred))
    print("RMSE:", np.sqrt(metrics.mean_squared_error(y_test, y_pred)))

    plt.figure()
    _
    plt.legend(loc="best")
    plt.show()
```

Visualization.

Figure 5-7. The regression process summarized.

5.4 Implementing the Application

5.4.1 Backend

This necessary section is implemented mainly by other team members in this project.

Our project uses a web application to show our project effect. So we choose front and back separation to make sure our work to be quicker.

Flask is a micro web framework written in Python.

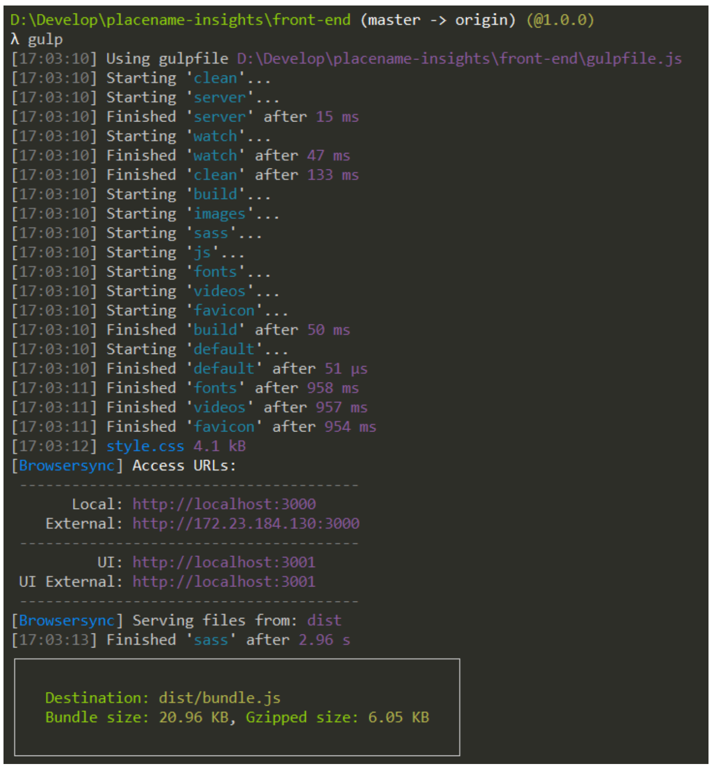
There are three parts in the backend code. First is the datasets directory, which is used to statistic some data. Next is the model directory, these model help us predict the location of city name inputted and do some statistics about the different countries.

5.4.2 Frontend

The frontend is written in extended HTML, SASS and ECMAScript6.

Gulp

Gulp is a toolkit for automating time-consuming tasks in the frontend development workflow.



```
D:\Develop\placename-insights\front-end (master -> origin) (@1.0.0)
λ gulp
[17:03:10] Using gulpfile D:\Develop\placename-insights\front-end\gulpfile.js
[17:03:10] Starting 'clean'...
[17:03:10] Starting 'server'...
[17:03:10] Finished 'server' after 15 ms
[17:03:10] Starting 'watch'...
[17:03:10] Finished 'watch' after 47 ms
[17:03:10] Finished 'clean' after 133 ms
[17:03:10] Starting 'build'...
[17:03:10] Starting 'images'...
[17:03:10] Starting 'sass'...
[17:03:10] Starting 'js'...
[17:03:10] Starting 'fonts'...
[17:03:10] Starting 'videos'...
[17:03:10] Starting 'favicon'...
[17:03:10] Finished 'build' after 50 ms
[17:03:10] Starting 'default'...
[17:03:10] Finished 'default' after 51 μs
[17:03:11] Finished 'fonts' after 958 ms
[17:03:11] Finished 'videos' after 957 ms
[17:03:11] Finished 'favicon' after 954 ms
[17:03:12] style.css 4.1 kB
[Browsersync] Access URLs:
 --------------------------------------
       Local: http://localhost:3000
    External: http://172.23.184.130:3000
 --------------------------------------
          UI: http://localhost:3001
 UI External: http://localhost:3001
 --------------------------------------
[Browsersync] Serving files from: dist
[17:03:13] Finished 'sass' after 2.96 s

    Destination: dist/bundle.js
    Bundle size: 20.96 KB, Gzipped size: 6.05 KB
```

Figure 5-8. The project gulp pipeline in progress.

extended HTML

Extended HTML is HTML with specific additional grammars. For instance, to include a inline SVG graphic resource in the HTML document, instead of inserting all SVG path declarations, one may use the following `include` keyword, which is an apparent clearer way:

```
@include("assets/arrow-left.svg")
```

The compiling is processed in a Gulp pipeline implemented as below:

```
gulp.task('html', ['images'], () => {
  return gulp.src('src/html/**/*.html')
   .pipe(plumber({ errorHandler: onError }))
   .pipe(include({ prefix: '@', basepath: 'src/' }))
   .pipe(htmlmin({ collapseWhitespace: true, removeComments: false }))
   .pipe(size(sizes))
   .pipe(gulp.dest('dist'))
})
```

SASS

Sass is a stylesheet language that's compiled to CSS. It allows variables, nested rules, mixins, functions, and more, all with a fully CSS-compatible syntax. Sass helps keep large stylesheets well-organized and makes it easy to share design within and across projects. Because that SASS is not natively supported by most of the browsers, before building into production, we need to compile the SASS code into native CSS.

The compiling is processed in a Gulp pipeline implemented as below:

```
// sass

const processors = [
  rucksack({ inputPseudo: false, quantityQueries: false }),
  prefixer({ browsers: 'last 2 versions' }),
  cssnano({ safe: true })
]

gulp.task('sass', () => {
  return gulp.src('src/sass/style.scss')
   .pipe(plumber({ errorHandler: onError }))
   .pipe(maps.init())
   .pipe(sass())
   .pipe(postcss(processors))
   .pipe(size(sizes))
   .pipe(maps.write('./maps', { addComment: false }))
   .pipe(gulp.dest('dist'))
})
```

ECMAScript6

ECMAScript6 is the sixth edition of ECMAScript language specification standard which is used in the implementation of JavaScript. Similar to SASS, codings in this language need to be compiled into an earlier version of JavaScript to avoid potential compatibility issues.

The compiling is processed in a Gulp pipeline implemented as below:

```
const read = {
  input: 'src/js/main.js',
  output: {
    sourcemap: true
 },
  plugins: [
    resolve({ jsnext: true, main: true }),
    commonjs(),
    babel({
      babelrc: false,
      presets: [
       [
          '@babel/preset-env', {
            modules: false,
            targets: {
              browsers: ['last 2 versions']
           }
         }
       ]
     ],
      plugins: [

     ]
   }),
    uglify(),
    filesize()
 ]
}
```

5.4.3 Multi-language Support

The Multi-language Support of this application is achieved with translater.js. This is a use of HTML comments page translation solution. For a small amount of static pages, this solution is more simple. it has no dependents, and lightweight. The usage of this library is demonstrated as below:

```
$ npm install translater.js
import 'translater.js';
```

Or manually download and link **translater.js** in your HTML, It can also be downloaded via unpkg:

```
          <p>
           Altogether six regression models are tested with our training
data:
            <!--{en}Altogether six regression models are tested with our
training data:-->
            <!--{zh}我们的训练数据共测试了六个回归模型：-->
            <!--{fr}Au total, six modèles de régression sont testés avec
nos données de formation:-->
            <ul>
              <li>Linear Regression</li>
              <li>Decision Tree</li>
              <li>K-neighbor</li>
              <li>Ada Boost Regression</li>
              <li>GBRTRegression</li>
              <li>Extra Tree</li>
            </ul>
          </p>
```

The method of switching languages via hyperlinks.

```
<a href="javascript:tran.setLang('default');">English</a>
<a href="javascript:tran.setLang('jp');">日本語</a>
<a href="javascript:tran.setLang('cn');">中文</a>
```

You can set the language parameter passed through URL.

```
http://127.0.0.1:9005/test/test.html?lang=jp
```

Main.js implementation in this project:

```
const main = () => {
  // A convenient i18n solution
  var tran = new Translater()
  tran.setLang('en')
  document.querySelectorAll('.trans').forEach((el) => {
    el.addEventListener('click', () => {
      let dimPanel = document.querySelector('.dim-panel');
      fadeIn(dimPanel)
      setTimeout(() => {
        tran.setLang(el.innerHTML)
        fadeOut(dimPanel)
      }, 500)
    })
  })

  // Detect browser language and set automatically
  switch(getFirstBrowserLanguage()) {
    case "zh-CN":
      tran.setLang('zh')
      break
    case "fr-FR":
      tran.setLang('fr')
      break
    default:
      tran.setLang('en')
  }
```

5.4.4 Integration

The frontend-backend integration process encountered problems of CORS/CORB. In order to eliminate safety controls during the local development testing, a copy of Chrome configuration is needed:

```
 "C:\Program Files (x86)\Google\Chrome\Application\chrome.exe" --
disable-web-security --disable-gpu --user-data-dir=~/chromeTemp
```

CHAPTER VI - SYSTEM TESTING

6.1 API Unit Testing

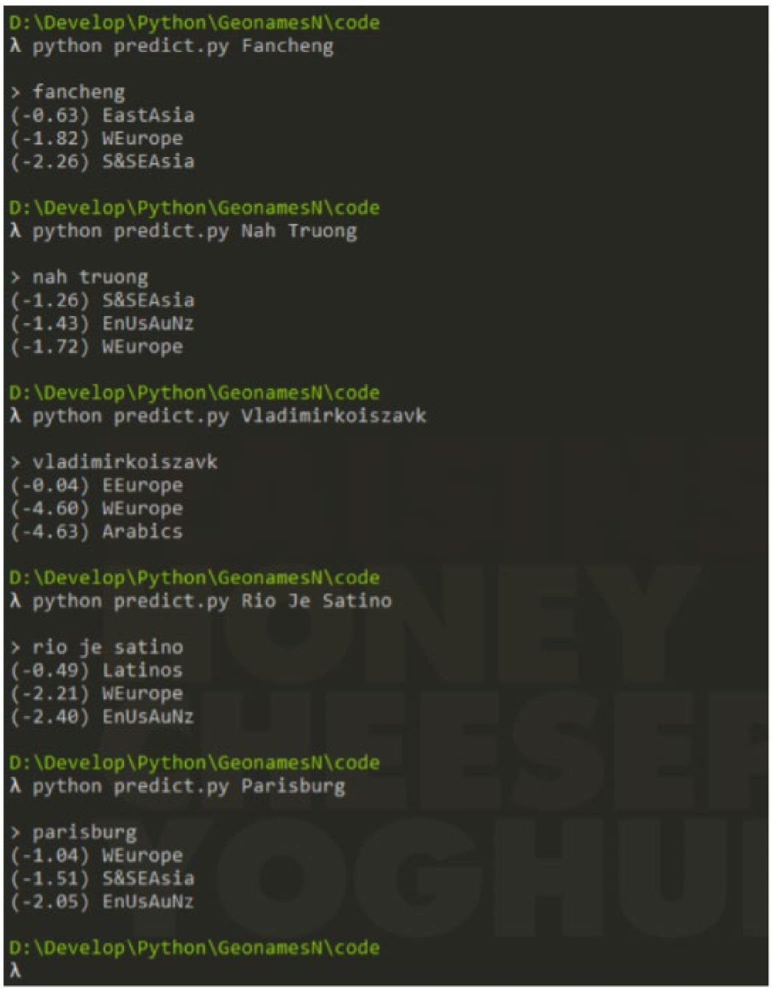
This necessary section is implemented mainly by other team members in this project.

For the testing of the whole program, we divide it into three modules to test, which are to test the existing real data, the data simulated according to the existing real data, the extreme situation and the data that may be reported incorrectly.

6.2 Machine Learning Testing

6.2.1 Sample Test

We fabricate several place names with typical geographic features (but not actual) to test the actual effect of RNN:



```
D:\Develop\Python\GeonamesN\code
λ python predict.py Fancheng

> fancheng
(-0.63) EastAsia
(-1.82) WEurope
(-2.26) S&SEAsia

D:\Develop\Python\GeonamesN\code
λ python predict.py Nah Truong

> nah truong
(-1.26) S&SEAsia
(-1.43) EnUsAuNz
(-1.72) WEurope

D:\Develop\Python\GeonamesN\code
λ python predict.py Vladimirkoiszavk

> vladimirkoiszavk
(-0.04) EEurope
(-4.60) WEurope
(-4.63) Arabics

D:\Develop\Python\GeonamesN\code
λ python predict.py Rio Je Satino

> rio je satino
(-0.49) Latinos
(-2.21) WEurope
(-2.40) EnUsAuNz

D:\Develop\Python\GeonamesN\code
λ python predict.py Parisburg

> parisburg
(-1.04) WEurope
(-1.51) S&SEAsia
(-2.05) EnUsAuNz

D:\Develop\Python\GeonamesN\code
λ
```

Figure 6-1. The Sample Tests.

We can notice that:

- Fancheng, a typical Chinese pinyin, classified by RNN under the East Asian label;
- Nah Truong, the Mandarin word after the Latinization of Vietnamese, classified under the South Asia/Southeast Asia label;
- Vladimirkoiszavk, a long list of Russian-style names, classified as Eastern Europe;
- Rio Je Satino, a typical Latin American/Western Portuguese place name, classified to Latin America;
- Parisburg, a hybrid of French and German names, is placed under the Western European directory.

Performance is basically in line with expectations. But this is only a perceptual understanding of RNN, we also need to quantitatively describe its performance.

6.2.2 Automated Test

We need a matrix of n*n to describe whether the city under each region is guessed (or guessed) as a city and study its laws and the reality it reflects. We can build a `counfusion` matrix to store the correct rate, standardize it, and draw the chart using the `matplotlib` library. code show as below:

```
 # Record the right or wrong in each type of guess in the confusion
matrix
 confusion = torch.zeros(n_categories, n_categories)
 n_confusion = 4000

 def randomChoice(l):
     return l[random.randint(0, len(l) - 1)]

 def randomTrainingExample():
     category = randomChoice(all_categories)
     line = randomChoice(category_lines[category])
     category_tensor = torch.tensor([all_categories.index(category)],
dtype=torch.long)
     line_tensor = lineToTensor(line)
     return category, line, category_tensor, line_tensor

 # Select a specific instance, guess and record one by one
 for i in range(n_confusion):
     category, line, category_tensor, line_tensor =
randomTrainingExample()
     guess, guess_i = guessOnce(line)
     category_i = all_categories.index(category)
     confusion[category_i][guess_i] += 1

 # Normalization
 for i in range(n_categories):
    confusion[i] = confusion[i] / confusion[i].sum()

# Set up plot
fig = plt.figure()
ax = fig.add_subplot(111)
cax = ax.matshow(confusion.numpy())
fig.colorbar(cax)

# Set up axes
ax.set_xticklabels([''] + all_categories, rotation=90)
ax.set_yticklabels([''] + all_categories)

ax.xaxis.set_major_locator(ticker.MultipleLocator(1))
ax.yaxis.set_major_locator(ticker.MultipleLocator(1))

plt.show()
```

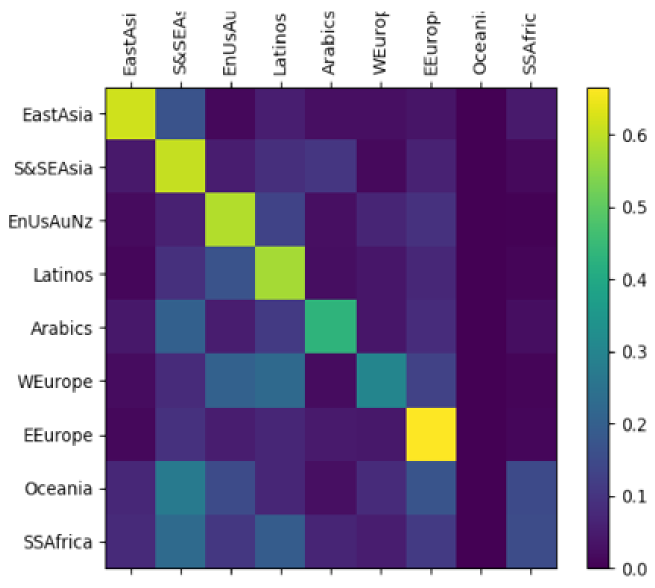


Figure 6-2. The confusion matrix for the first model.

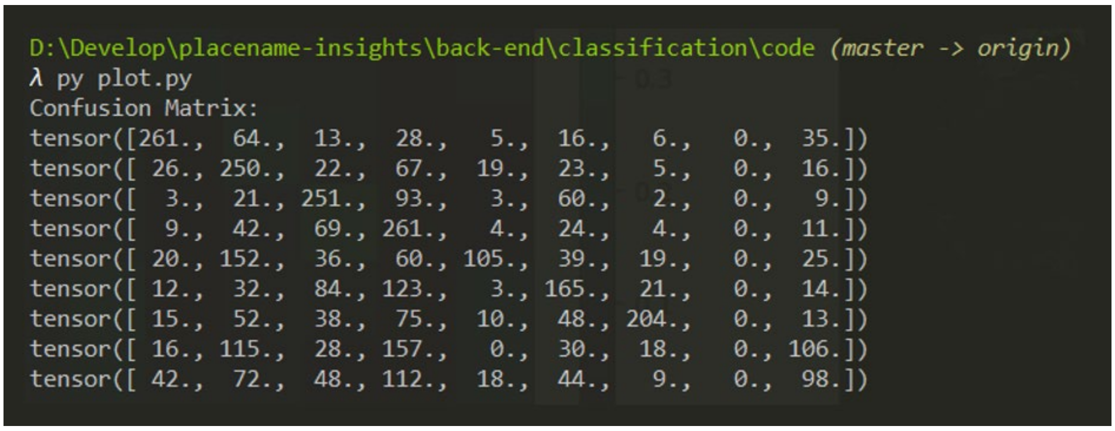
It can be seen from the plot that, each horizontal row represents the correct classification, and each series represents the result label calculated by the RNN. For example, the upper left corner corresponds to the probability that the East Asian city is (correctly classified as an East Asian city).

It can be seen that the bright spots on the diagonal mean that for most regions, **the RNN we train can be classified with a higher accuracy rate; but the correct rate for the classification of city names in Oceania and sub-Saharan Africa is very low.** This result is influenced by a certain realistic background and data factors. See the following section for specific considerations.

6.2.3 Discussion on Test Results

"Papua New Guinea is the most spoken language in the world, with 839 languages, mostly indigenous languages, not dialects. This figure is nearly three times the sum of European languages. As the country with the most languages in the world, PNG is proud, and the diversity of languages means that diverse cultural and humanistic ecology can be passed down and maintained."

—— Wang Xinqiang, Global Times



```
D:\Develop\placename-insights\back-end\classification\code (master -> origin)
λ py plot.py
Confusion Matrix:
tensor([261.,  64.,  13.,  28.,   5.,  16.,   6.,   0.,  35.])
tensor([ 26., 250.,  22.,  67.,  19.,  23.,   5.,   0.,  16.])
tensor([  3.,  21., 251.,  93.,   3.,  60.,   2.,   0.,   9.])
tensor([  9.,  42.,  69., 261.,   4.,  24.,   4.,   0.,  11.])
tensor([ 20., 152.,  36.,  60., 105.,  39.,  19.,   0.,  25.])
tensor([ 12.,  32.,  84., 123.,   3., 165.,  21.,   0.,  14.])
tensor([ 15.,  52.,  38.,  75.,  10.,  48., 204.,   0.,  13.])
tensor([ 16., 115.,  28., 157.,   0.,  30.,  18.,   0., 106.])
tensor([ 42.,  72.,  48., 112.,  18.,  44.,   9.,   0.,  98.])
```

Figure 6-3. The confusion matrix in numbers.

For the reasons for the low recognition rate of the two geographic regions of Oceania and SSAfrica in the confusion matrix, we discuss the possible factors. We believe that the countries in these two regions are scattered first, resulting in heavy cultural isolation. For example, in Papua New Guinea, the largest country under the Oceania label, the local language is extremely complex and diverse; in addition, the lack of a large enough city in Oceania Island countries and insufficient training samples have made RNN unable to adequately determine the characteristics of such city names, which is also an important reason.

In an attempt to demonstrate our guess, we modified the logic of randomly selecting rows of data during part of the training process: randomly selecting a tuple from the “all rows of data” and changing to randomly selecting a geographic region from “all geographic regions”. Then randomly select the tuple from this area. This logic causes the data samples drawn by each region to be substantially equal:

```
# Before Modification
category = randomChoice(all_categories)
line = randomChoice(category_lines[category])

# After Modification
tuple = randomChoice(all_lines)
line = tuple[0]
```

```
    category = tuple[1]
```

The revised results show that Oceania Island has become one of the regions with the highest correct recognition rate, because the few cities in the catalog have participated in training almost all times and have had a considerable impact on RNN. This performance demonstrates our guess.

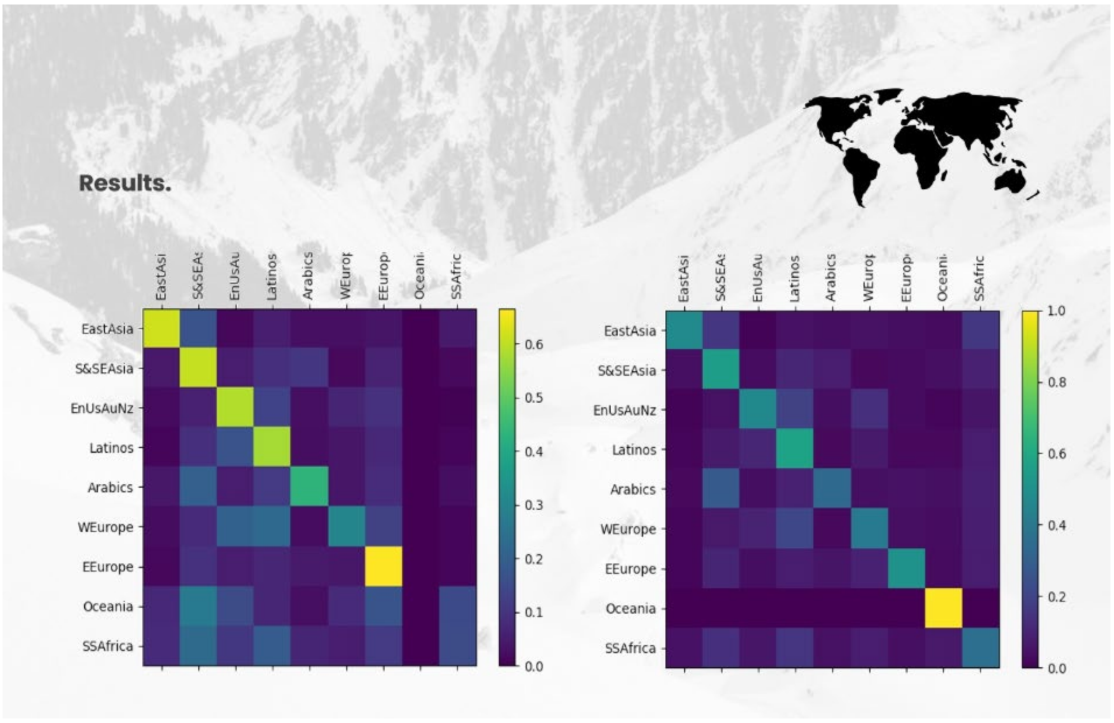


Figure 6-4. Comparison between the two confusion matrices.

Returning to the results of the first experiment, the geographical recognition rate of Eastern Europe, East Asia, Southeast Asia and other regions is higher in the areas that can be correctly identified. **This is also in line with our intuition, because the spelling of Russian Latinization is quite high.** Identification (high consonant letters/vowels, common sk-ya suffixes, etc.), while Chinese Pinyin and Southeast Asian languages are mostly analytical, and monosyllabic ideograms are commonly used, resulting in higher vowels. Probability of letters and distinct syllable intervals are very easy to capture by RNN.

CHAPTER VII - DEPLOYMENT & CONTINUOUS INTEGRATION

This necessary chapter is implemented mainly by other team members in this project.

7.1 Deployment

In order for the project to be accessible online and closer to the production environment during the testing process, we need to deploy the front end and back end of the entire project to a remote server for testing and access.

Using the 2GB memory server that AWS deployed in Paris, our deployment process went smoothly—every command will get a quick response and run results.

We used the latest LTS version (18.04) of Ubuntu system as the operating system, Ubuntu is a free and open-source Linux distribution based on Debian. Because it is really suitable for our project. The system's own Python3 environment is exactly what we need, and the powerful apt tool allows us to easily install the latest versions of the various tools we need, such as pip3 / NodeJs / Npm / Nginx / Jenkins.

Pip is a package-management system used to install and manage software packages written in Python. Many packages can be found in the default source for packages and their dependencies — Python Package Index (PyPI).

With pip, we can install a required Python environment with a single command at a time, such as Tensorflow / Pandas / NumPy / Flask, so that our online environment will be as good as the development environment.

Node.js is an open-source, cross-platform JavaScript run-time environment that executes JavaScript code outside of a browser. Node.js lets developers use JavaScript to write command line tools and for server-side scripting—running scripts server-side To produce dynamic web pagecontent before the page is sent to the user's web browser.

NodeJs is used because the front-end project uses this environment. It not only provides the Npm package management tool, but also provides a running environment for many useful tools, such as gulp used in this project, which can provide automatic image compression and CSS format conversion. And other practical features.

Nginx is a web server which can also be used as a reverse proxy, load balancer, mail proxy and HTTP cache.

With Nginx, we can use reverse proxy to put the front and back projects deployed on different ports and different directories under a unified domain name. This avoids the server opening more ports to the outside world, enhances security, and by the way, solves the Cross-domain issues.

When the project is in the middle and late stages, the framework has been basically laid, and there will be no major changes. The main purpose of redeployment is to test the changes brought by a few code changes. If you manually deploy the project every time, it will lead to a lot of homogenization. And meaningless work, so at this time we deployed Jenkins on the server as a persistent integration application to avoid the huge workload caused by each manual deployment.

7.2 Continuous integration (CI)

Jenkins is an open source automation server written in Java. Jenkins helps to automate the non-human part of the software development process, with continuous integration and facilitating technical aspects of continuous delivery. It is a server-based system that runs in servlet containers such as Apache Tomcat. With Jenkins, we can finally omit the same deployment command every time, replace it with a web page that accesses Jenkins and click the Build button to complete the deployment.

But this is not enough. It still requires us to visit the webpage to build the project when we want to redeploy the project. It does not achieve complete automation and brings unnecessary workload.

7.2.1 Additional Thinking

Although our server's hardwares are pretty efficient as a student server, it still takes about 2.5 minutes for each deployment, which is not normal even for Jenkins, a server-critical tool. After carefully observing the console output during the build process, I found that the most time-consuming command was `npm install`, but because the backend developer did not include the files generated after the build when writing the gitignore file, each build would need to be The entire local repository is emptied and re-clone, so `npm install` cannot be omitted and should be improved in later development of the project.

CHAPTER VIII - SYSTEM INSTALLATION & USER MANUAL

8.1 System Installation

Meaningless topic. Skipped.

8.2 User Manual

To use the software released by this project, since the final work is based on the web platform, only the browser and the available network are needed, without additional installation process, environment configuration and learning. The web interface is simple and easy to use, and friendly to first time users.

A few notes can be summarized as follows:

8.2.1 Running the Application

Generally, one can just type pn-i.club on an arbitrary browser to initiate our application. **However, considering the cost of running the server and purchasing the domain name, we only deployed it on the Internet on the day of the project's live show. Currently, this app may not be directly available through the web because our server has expired.** However, if you want, you can always run our application by setting up our nodejs and python servers locally and listening to html files.

8.2.2 Setting the Language

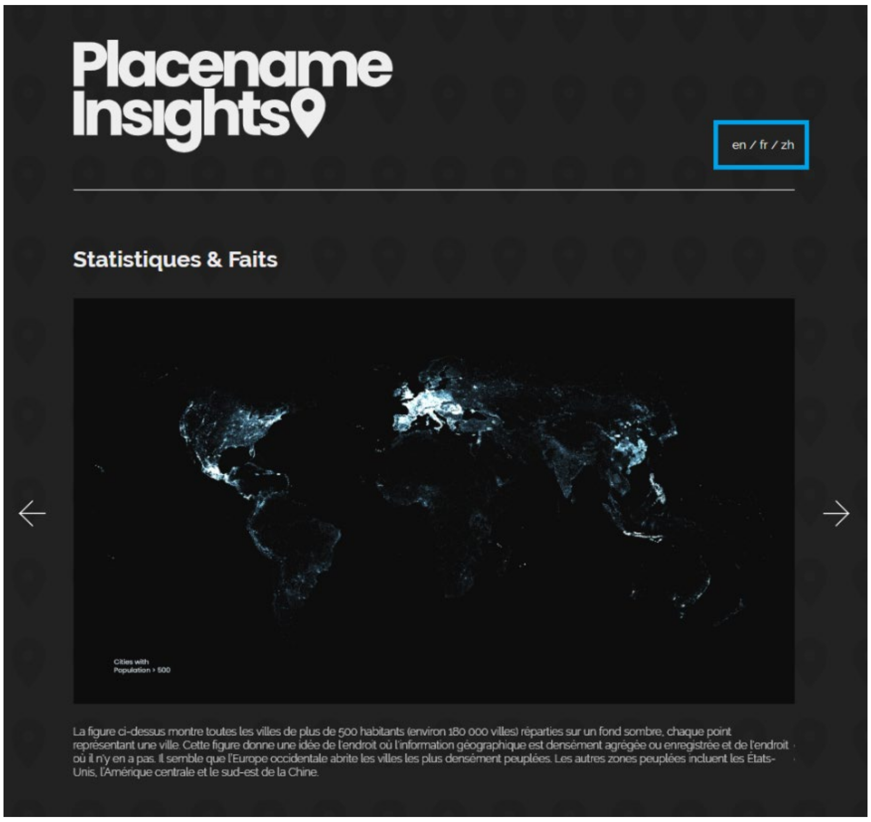
Our application has been fully internationalized, allowing it to switch freely between Chinese, English and French. Notice the toggle button in the top right corner of the page.

In order to save user time and optimize the user experience, we have written the logic to determine the user's system language at the beginning of the application launch and automatically set the system language. **Therefore, if your system language or browser preferred language is Chinese, our application interface may first be displayed in Chinese**. Switch at any time.

8.2.3 Navigating The Lightbox Gallery

At the top of our program page, a series of slides are displayed to demonstrate the visualization work we completed during the experiment and the corresponding analysis summary.

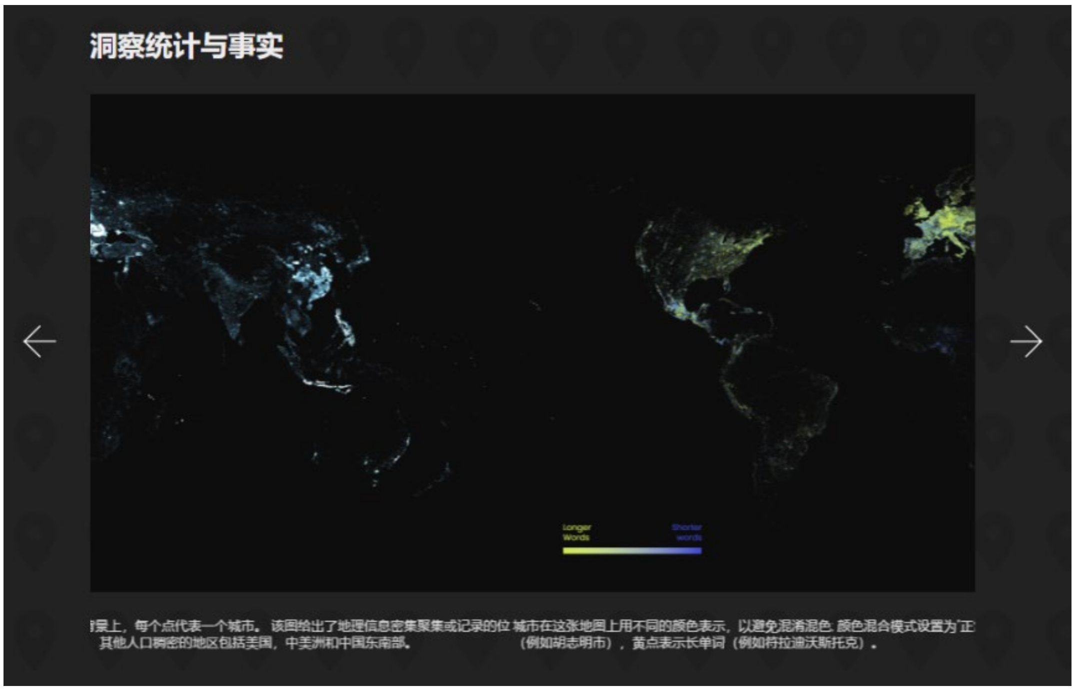
The slide switch function is implemented by listening to mouse actions or touch screen actions. You can choose to drag the picture to switch to the next one, slide the picture on the touch screen, or click the left and right buttons with the mouse to achieve the same effect.



Statistiques & Faits

La figure ci-dessus montre toutes les villes de plus de 500 habitants (environ 180 000 villes) réparties sur un fond sombre, chaque point représentant une ville. Cette figure donne une idée de l'endroit où l'information géographique est densément agrégée ou enregistrée et de l'endroit où il n'y en a pas. Il semble que l'Europe occidentale abrite les villes les plus densément peuplées. Les autres zones peuplées incluent les États-Unis, l'Amérique centrale et le sud-est de la Chine.

Figure 8-2. The multiple language support.

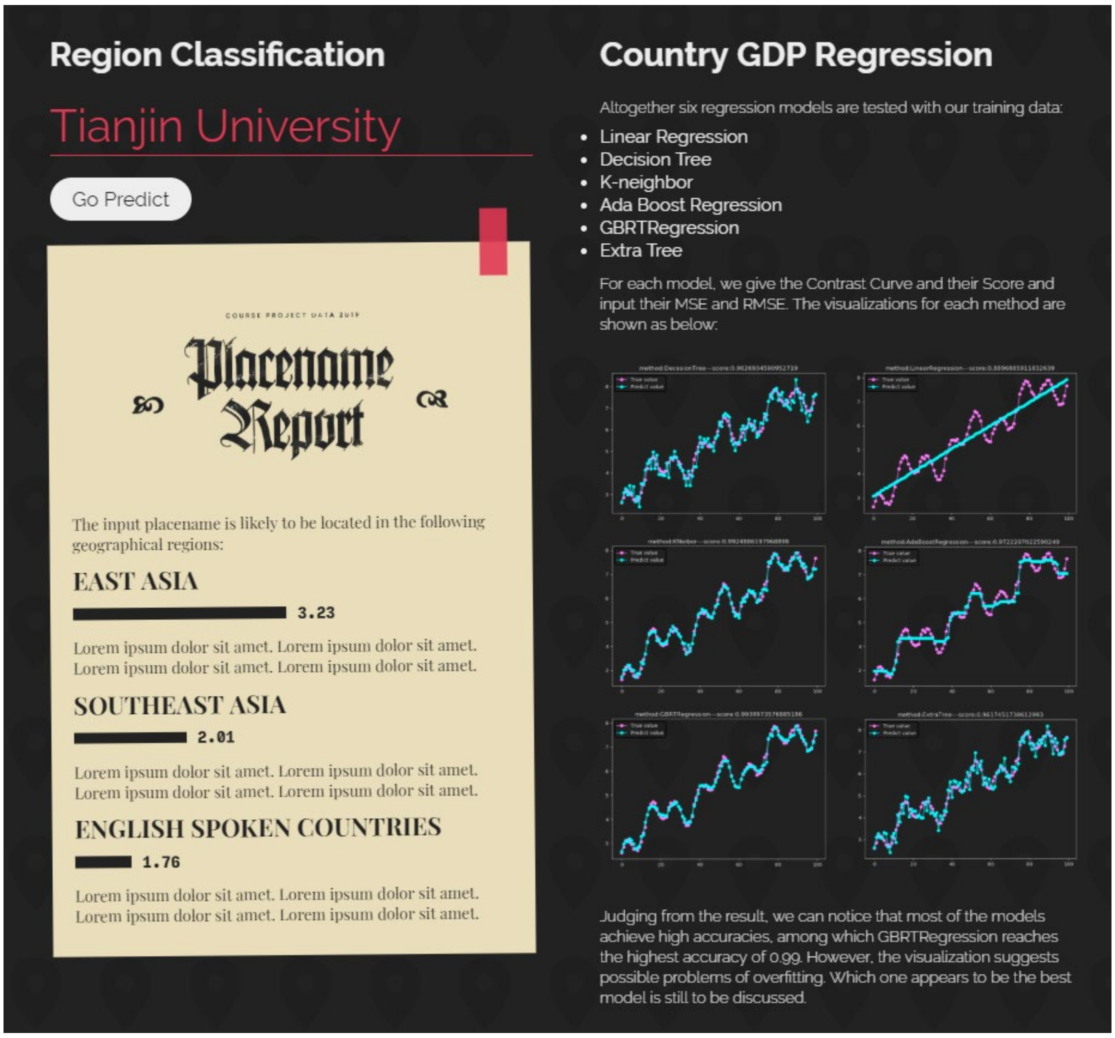


洞察统计与事实

景上，每个点代表一个城市。该图给出了地理信息密集聚集或记录的位 城市在这张地图上用不同的颜色表示，以避免混淆混色. 颜色混合模式设置为'正
其他人口稠密的地区包括美国，中美洲和中国东南部。 (例如胡志明市)，黄点表示长单词（例如特拉迪沃斯托克）。

Figure 8-3. The lightbox carousel.

8.2.4 Predicting Place Names


Region Classification

Tianjin University

Go Predict

COURSE PROJECT DATA 2019

Placename Report

The input placename is likely to be located in the following geographical regions:

EAST ASIA

3.23

Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet.

SOUTHEAST ASIA

2.01

Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet.

ENGLISH SPOKEN COUNTRIES

1.76

Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet.

Country GDP Regression

Altogether six regression models are tested with our training data:

- Linear Regression
- Decision Tree
- K-neighbor
- Ada Boost Regression
- GBRTRegression
- Extra Tree

For each model, we give the Contrast Curve and their Score and input their MSE and RMSE. The visualizations for each method are shown as below:

Judging from the result, we can notice that most of the models achieve high accuracies, among which GBRTRegression reaches the highest accuracy of 0.99. However, the visualization suggests possible problems of overfitting. Which one appears to be the best model is still to be discussed.

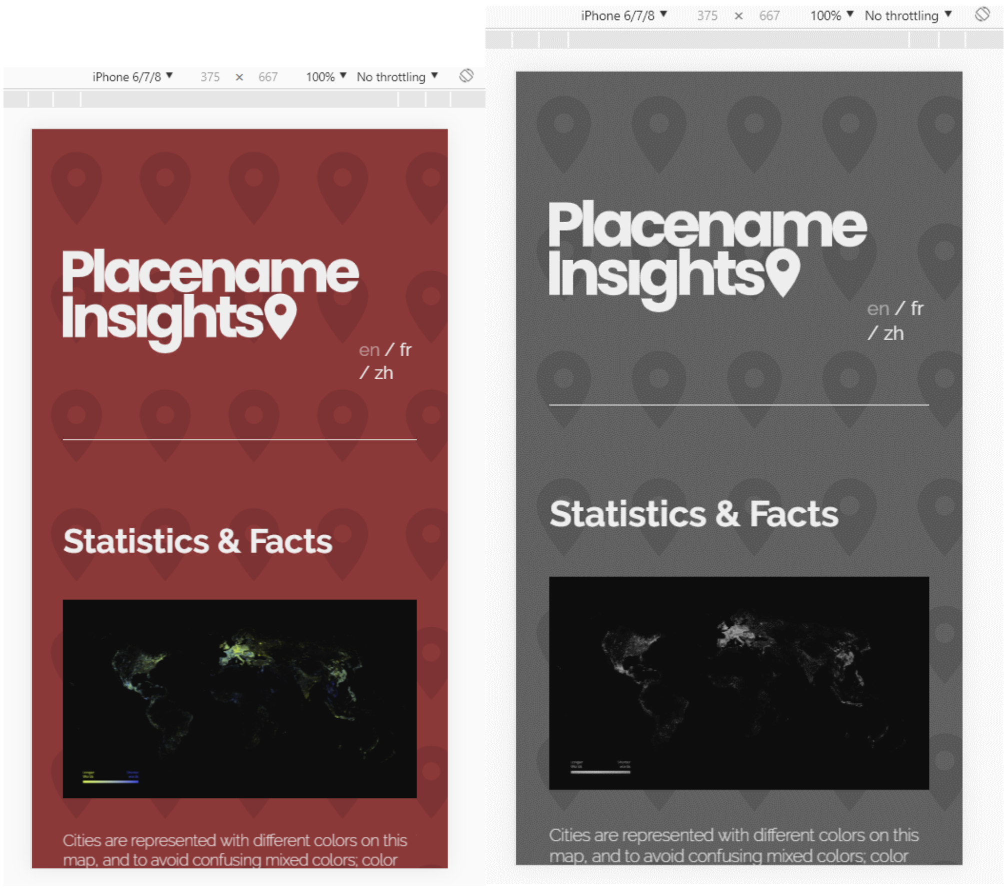
Figure 8-4. The location prediction module.

Place name prediction is the main function of our application. The interface is very intuitive, in this part, you only need to enter a string representing the place name, you can request the prediction results.

The top three names of the geographical regions with the highest probability are shown in the report, and a brief summary of the general characteristics of the geographical names of the three regions is given. You can find that our cost function calculation value is converted to a bar chart by a certain mapping to represent the probability.

8.2.5 Responsive Design: Use on Multiple Devices

Our application implements a responsive layout design, so it is compatible with electronic devices of various sizes and interactive media such as mobile phones, tablets and desktop computers. Part of the layout effect is shown below:


iPhone 6/7/8 ▾ 375 × 667 100% ▾ No throttling ▾

Placename Insights

en / fr / zh

Statistics & Facts

Cities are represented with different colors on this map, and to avoid confusing mixed colors; color

iPhone 6/7/8 ▾ 375 × 667 100% ▾ No throttling ▾

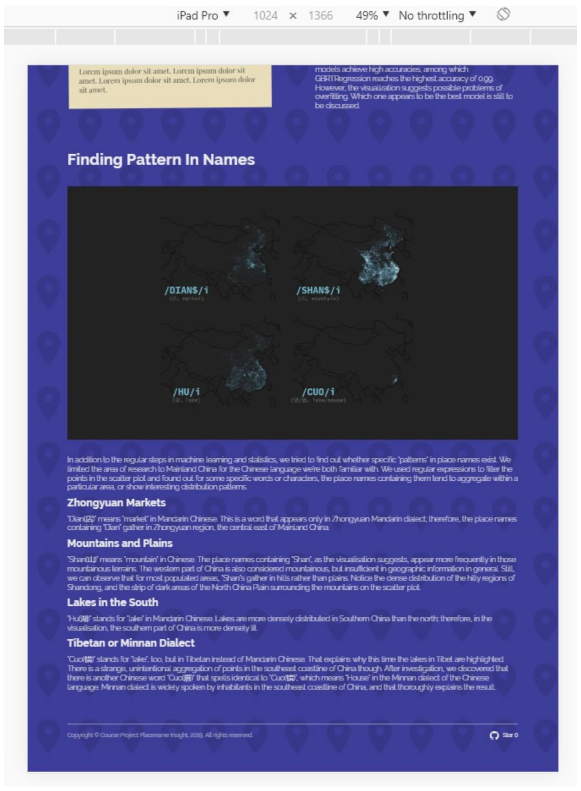
Placename Insights

en / fr / zh

Statistics & Facts

Cities are represented with different colors on this map, and to avoid confusing mixed colors; color

Figure 8-5. Responsive layout on mobile devices.


iPad Pro ▾ 1024 × 1366 49% ▾ No throttling ▾

Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet.

models achieve high accuracies, among which GBRTRegression reaches the highest accuracy of 0.99. However, the visualization suggests possible problems of overfitting. Which one appears to be the best model is still to be discussed.

Finding Pattern In Names

In addition to the regular steps in machine learning and statistics, we tried to find out whether specific "patterns" in place names exist. We limited the area of research to Mainland China for the Chinese language we're both familiar with. We used regular expressions to filter the points in the scatter plot and found out for some specific words or characters, the place names containing them tend to aggregate within a particular area, or show interesting distribution patterns.

Zhongyuan Markets

"Dian(店)" means "market" in Mandarin Chinese. This is a word that appears only in Zhongyuan Mandarin dialect; therefore, the place names containing "Dian" gather in Zhongyuan region, the central east of Mainland China.

Mountains and Plains

"Shan(山)" means "mountain" in Chinese. The place names containing "Shan", as the visualisation suggests, appear more frequently in those mountainous terrains. The western part of China is also considered mountainous, but insufficient in geographic information in general. Still, we can observe that for most populated areas, "Shan"s gather in hills rather than plains. Notice the dense distribution of the hilly regions of Shandong, and the strip of dark areas of the North China Plain surrounding the mountains on the scatter plot.

Lakes in the South

"Hu(湖)" stands for "lake" in Mandarin Chinese. Lakes are more densely distributed in Southern China than the north; therefore, in the visualisation, the southern part of China is more densely lit.

Tibetan or Minnan Dialect

"Cuo(错)" stands for "lake", too, but in Tibetan instead of Mandarin Chinese. That explains why this time the lakes in Tibet are highlighted. There is a strange, unintentional aggregation of points in the southeast coastline of China though. After investigation, we discovered that there is another Chinese word "Cuo(厝)" that spells identical to "Cuo(错)", which means "House" in the Minnan dialect of the Chinese language. Minnan dialect is widely spoken by inhabitants in the southeast coastline of China, and that thoroughly explains the result.

Copyright © Course Project Placename Insight, 2019. All rights reserved.

Star 0

Figure 8-6. Responsive layout on tablets/iPads.

CHAPTER IX - PROJECT SUMMARY REPORT

9.1 Overall Conclusion

Place names around the world have a subtle and close relationship with their location. This is usually because different regions have different languages and writing systems, hence the different spelling patterns.

Characteristics of place names differ due to geographic location changes. To a certain extent, the changes are both continuous and discrete. They're continuous because place names are similar in neighboring regions. For instance, throughout the Europe continent, the proportion of consonants in their spelling rises from the Mediterranean to the north. Another example, inside China, the unified country, the place names in the northwest are more "Arabic" than those in the east.

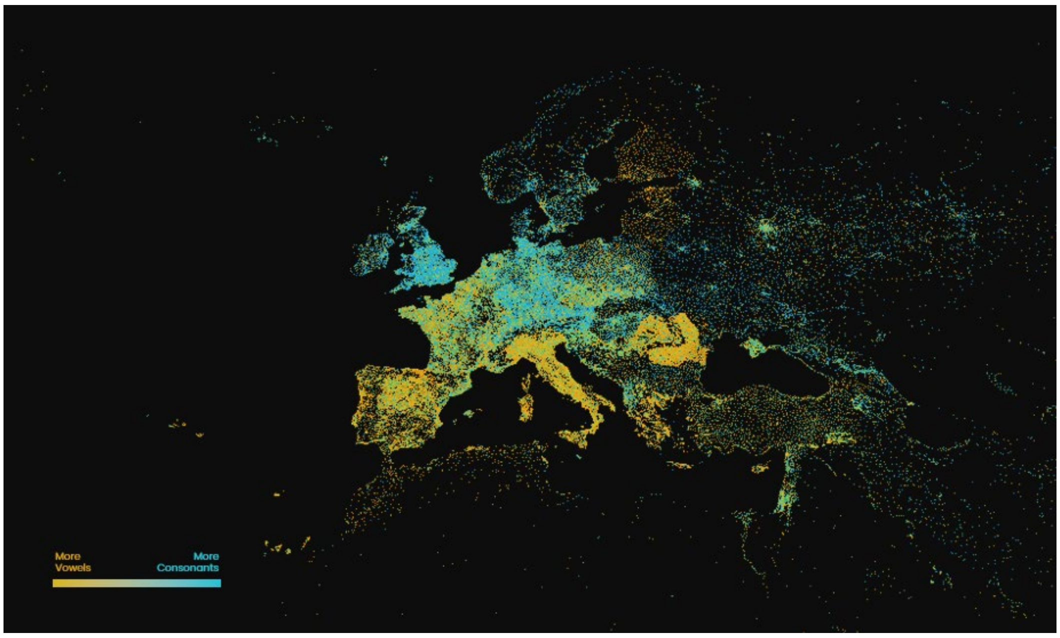


Figure 9-1. The proportion of consonants throughout the Europe continent.

The discrete character of place names, on the other hand, is usually caused by geographical barriers, political boundaries, and cultural divisions. For example, in Spain and Argentina, two countries that locates vastly different, have similar place names because of historical colonial activities. Like the border between China and Vietnam, they share similar cultures, but the official language on one side is In Chinese, while Vietnamese on the other side - the two writing systems, therefore, have different Latin transcription standards, resulting in the latter's place names usually being split into many relatively short words. This "discrete" character is caused by political reasons.

Some geographical regions have quite obvious place name patterns. For instance, in East Europe, especially Russia, place names are usually constructed by a single long word, which makes them relatively easier to be recognized. Some Slavic suffixes like “-sk” are commonly seen here. Some geographical areas, such as sub-Saharan Africa and Oceania island countries, the culture and languages are so diverse there causing the place names are much more challenging to identify.

9.2 Visualization Summary

9.2.1 Populated Cities Distribution

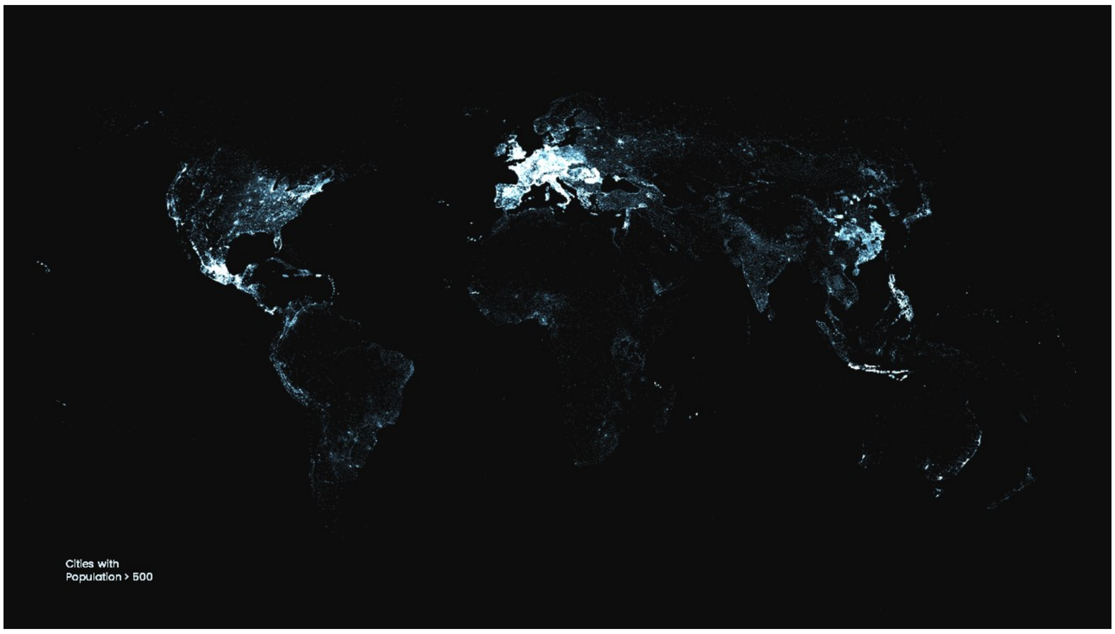


Figure 9-2. All cities with population more than 500.

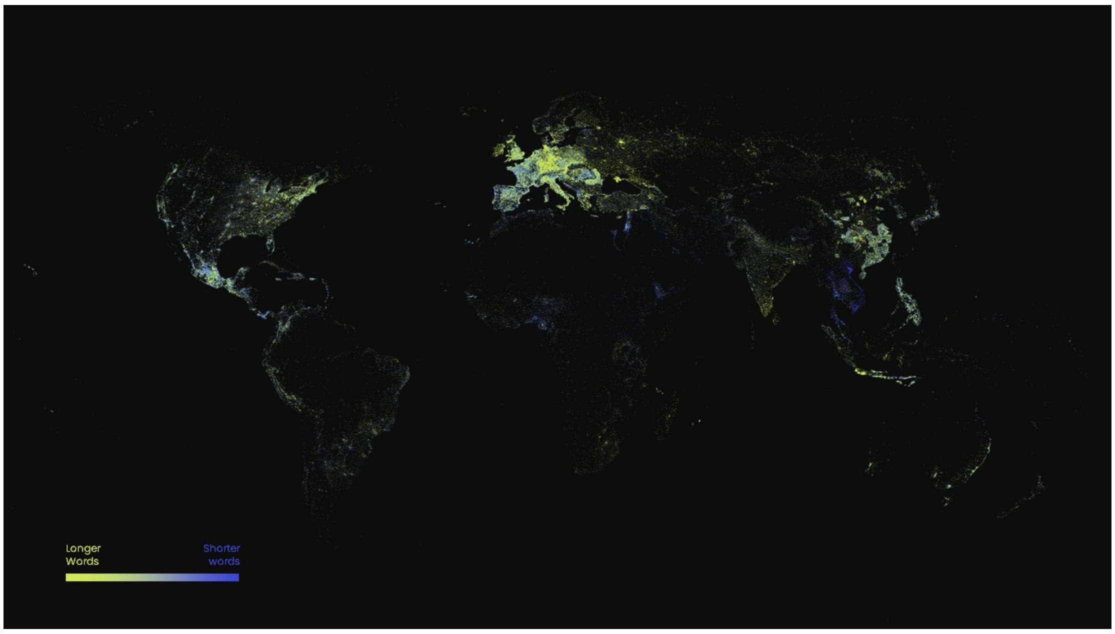


Figure 9-3. Cities with longer words.

The figure above shows all cities with population more than 500 (total ~180,000 cities) scattered onto a dark background, with each point represents a city. This figure gives a sense of where geographic information is densely aggregated or recorded and where there are not. It seems that western European area holds the most densely distributed populated cities. Other populated area includes the USA, Central America and Southeast China.

9.2.2 Cities with Longer Words

Cities are represented with different colors on this map, and to avoid confusing mixed colors; color *blend modes* are set to Normal, i.e. no blending. In this figure, blue dot means a city that has short words in its name (e.g. Ho Chi Minh City), and yellow dot indicates long words (e.g. Vladivostok).

To design a proper `words_length -> color` projection function, we firstly investigated the distribution of word lengths. The median word length is around six letters, with a minimum of 1 letter and a maximum of over 20 letters. This is like a normal distribution with *skewed* (or imbalanced) two sides. *Log-normal distribution*, in this case, fits the model.

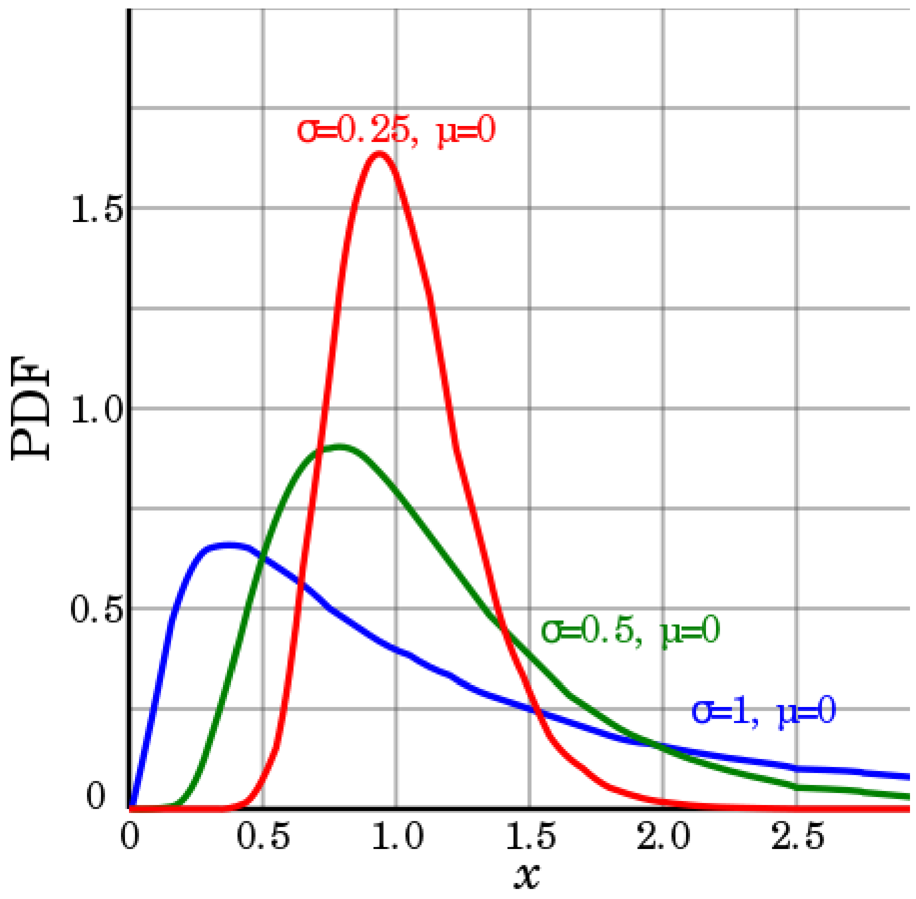


Figure 9-4. The PDF of the distribution model.

Above is its PDF (Probability density function) plotted. To transform the distribution into a color projection function, we need its CDF (Cumulative distribution function) expressions:

```
 CDF(x, \mu, \sigma) = \frac12 + \frac12\operatorname{erf}\Big[\frac{\ln x-
\mu}{\sqrt{2}\sigma}\Big]
```

The corresponding part of the code is implemented as below:

```
     function erf(x) {
         let m = 1.00;
         let s = 1.00;
         let sum = x * 1.0;
         for (let i = 1; i < 50; i++) {
             m *= i;
             s *= -1;
             sum += (s * Math.pow(x, 2.0 * i + 1.0)) / (m * (2.0 * i +
1.0));
        }
         return 2 * sum / Math.sqrt(3.14159265358979);
    }

     function logNormalCDF(x, mu, sigma) {
         let par = (Math.log(x) - mu) / (Math.sqrt(2) * sigma)
         return 0.5 + 0.5 * erf(par)
    }

     const projectColor = (x) => Math.round(logNormalCDF(x/5, 0, 1)*255)
```

Moreover, despite its ugliness, the result color is concatenated from strings:

```
 context.fillStyle = 'rgb(' + projectColor(wordLength) + ',' +
projectColor(wordLength) + ',' + (255 - projectColor(wordLength)) + ')'
```

9.2.3 Cities with More Words

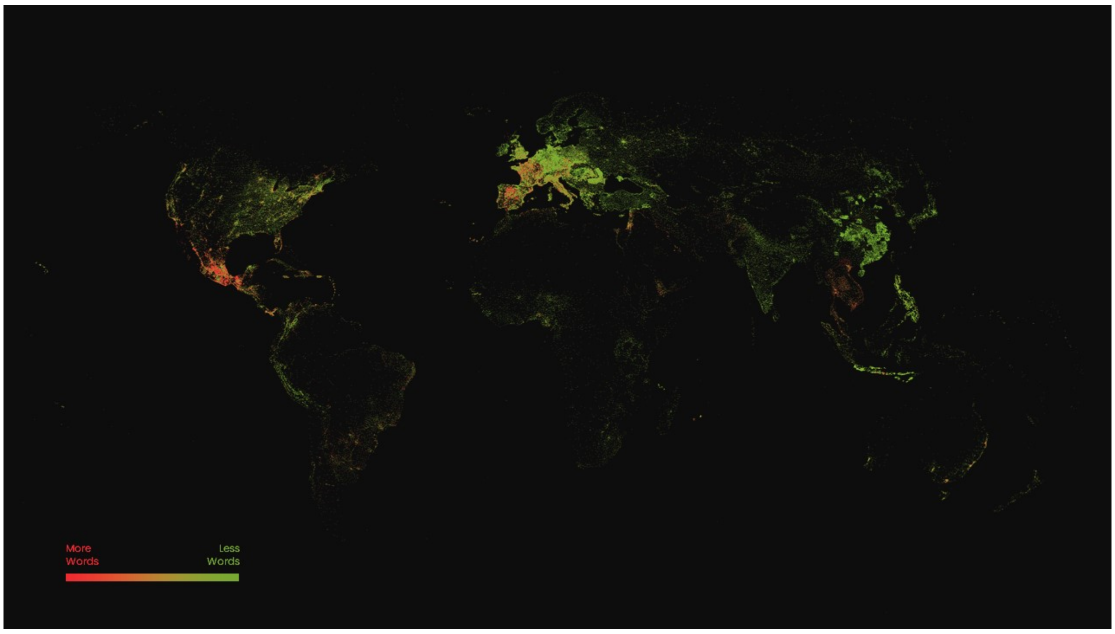


Figure 9-5. Cities with more words.

The idea behind this image is similar to the previous plot, whereas a red dot indicates a city name consists of many words, and a green dot indicates the contrary.

9.2.4 Cities with Higher Vowel-Consonant Ratio

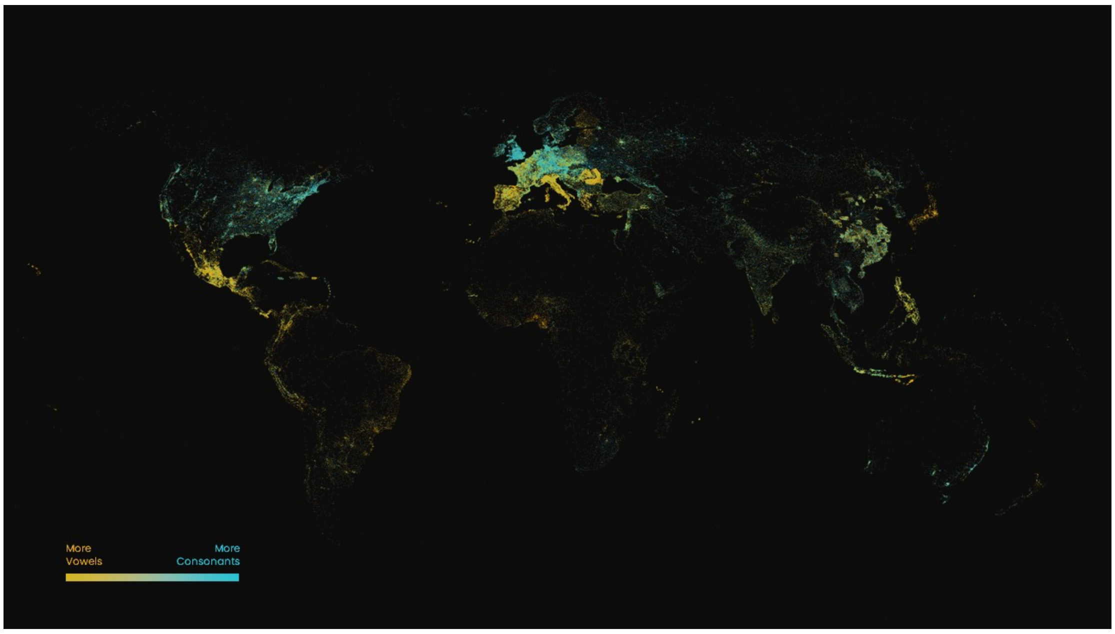


Figure 9-6. An experiment with vowel-consonant ratio.

We experimented with vowel-consonant ratio in this map. So if you learned the alphabet, you'll know that vowels only include "aeiou" five letters, and all other letters are consonants. So more vowels in a word is likely to make the pronunciation "softer". Like "Hia-ou-ei-oi", that's obviously a word with more “aeiou”s, so it sounds softer. A word with a lower vowel-consonant ratio, which means more consonants in their spelling, usually sounds hard. Like this “Psktibk”.

So We can say that on this map, the yellow part indicates the place names that sounds “softer”, and blue indicates “harder”. We can see that cities in Japan sounds really soft, in Mexico it’s soft, in France, Italy and Spain, soft, but in America, England, German, they sound really hard.

This is an example of how spelling of a place name affects its pronunciation.

9.2.5 Experimenting with Placenames in China

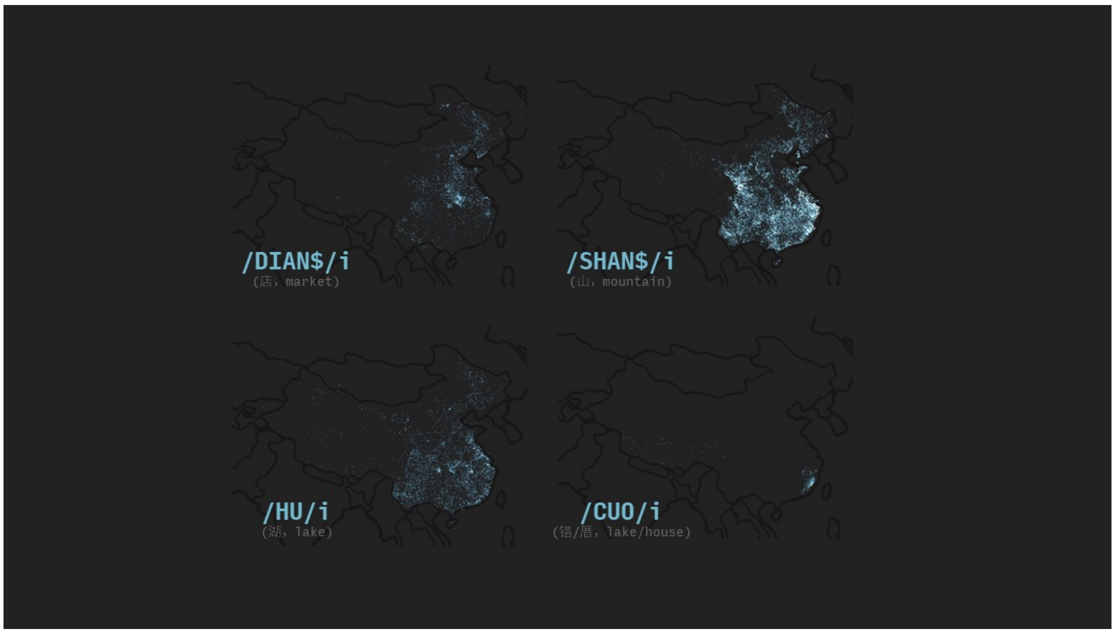


Figure 9-7. Filtering certain patterns in China place names.

In addition to the regular steps in machine learning and statistics, we tried to find out whether specific "patterns" in place names exist. We limited the area of research to Mainland China for the Chinese language we're both familiar with. We used regular expressions to filter the points in the scatter plot and found out for some specific words or characters, the place names containing them tend to aggregate within a particular area, or show interesting distribution patterns.

Zhongyuan Markets

"Dian(店)" means "market" in Mandarin Chinese. This is a word that appears only in Zhongyuan Mandarin dialect; therefore, the place names containing "Dian" gather in Zhongyuan region, the central east of Mainland China.

Mountains and Plains

"Shan(山)" means "mountain" in Chinese. The place names containing "Shan", as the visualisation suggests, appear more frequently in those mountainous terrains. The western part of China is also considered mountainous, but insufficient in geographic information in general. Still, we can observe that for most populated areas, "Shan"s gather in hills rather than plains. Notice the dense distribution of the hilly regions of Shandong, and the strip of dark areas of the North China Plain surrounding the mountains on the scatter plot.

Lakes in the South

"Hu(湖)" stands for "lake" in Mandarin Chinese. Lakes are more densely distributed in Southern China than the north; therefore, in the visualisation, the southern part of China is more densely lit.

Tibetan or Minnan Dialect

"Cuo(错)" stands for "lake", too, but in Tibetan instead of Mandarin Chinese. That explains why this time the lakes in Tibet are highlighted. There is a strange, unintentional aggregation of points in the southeast coastline of China though. After investigation, we discovered that there is another Chinese word "Cuo(厝)" that spells identical to "Cuo(错)", which means "House" in the Minnan dialect of the Chinese language. Minnan dialect is widely spoken by inhabitants in the southeast coastline of China, and that thoroughly explains the result.

9.3 Regional Features Summary on Spelling

The writing system in **East Asia** is usually ideographic, which generates clear syllable boundaries in its place names. Some common spelling patterns, including “-eng”, “-ang” can be seen.

The features in **Southeast Asia and South Asia** placenames is somewhat similar to that of East Asia. However, the average word length is dramatically shorter because each ideograph represents one word. The “Syllable boundary” in East Asia placename words simply becomes whitespaces here.

Place names in **Africa** (excluding the northern Arabic cultural region) are diverse in characteristics, making them harder to recognize. We can occasionally notice a Europe impact, especially France, on their place names.

Place names in **Oceania** are diverse in characteristics and meanwhile short in collected data. This makes them nearly impossible to be classified.

Place names in **West Europe** typically demonstrate a low vowel-consonant ratio, which means the pronunciation sounds “harder”. Some common suffixes like “-burg”, “-eaux” can be observed.

Place names in **East Europe** are usually constructed by a single long word, which makes them relatively easier to be recognized. Some Slavic suffixes like “-sk” are commonly seen here. Place names from English-spoken countries (North America, Australia, New Zealand, and the UK) show features in the English language.

Place names in **Latin Region**, including Latin America, Spain, and Portugal, are made up of relatively short words. Some common prefixes like “de”, “le”, “san” can be observed.

Place names in the **Arabic Cultural Region** usually suggest some unique patterns, including "Al", "Ah", and "-j". Some difficultly pronounced combinations of letters are detected.