上海交通大學

SHANGHAI JIAO TONG UNIVERSITY

学士学位论文

BACHELOR’S THESIS

论文题目：Traffic Prediction for Urban Planning

学生姓名：许方正

学生学号：5120309679

专　　业：计算机科学与技术

指导教师：朱其立教授

学院(系)：电子信息与电气工程学院

城市规划中的道路交通预测

摘要

在本文我们尝试通过实现两个目标来解决城市规划中的道路交通预测问题。其中之一是利用机器学习算法提取特征，训练模型，并预测交通状况。另一个则是实现了一个预测系统，使得永和可以编辑、浏览地图，并预测交通，将结果可视化展现在原地图界面上。我们介绍了许多我们使用的不同数据来源，包括开放地图数据 OpenStreetMap，还有百度地图的实时交通数据和兴趣点（POI）信息，百度地图不是开放的，我们通过爬取、解析这些栅格图像数据获得信息。接下来我们讨论了有关特征工程的一般想法，并介绍了预测使用的三类特征。利用外界第三方有关地点热度的数据源，并使用了 Affinity Propagation 算法聚类，A*搜索算法等等来挖掘提取某些非本地的特征。随后介绍了模型选择比较和训练，测试和评估等。我们使用多分类的支持向量机（SVM）分类器。最后，我们详细介绍了我们预测系统从后端接口到前端界面的设计与实现。我们还证明了我们的特征和模型以及系统都有能力以较好的准确度，以道路网络，兴趣点，其他空间地理数据，以及时空数据和非空间数据作为输入，预测未来交通状况。

关键词： 交通预测，城市规划，支持向量机（SVM），Affinity Propagation，A*搜索算法，OpenStreetMap，时空数据挖掘

Traffic Prediction for Urban Planning

ABSTRACT

We try to solve the traffic prediction in urban planning by achieving two tasks in this thesis, one is the machine learning algorithms that extract the features, train the model and predict the traffic situation with given input features; while the another is the implementation of a backend system and web application that enables user to edit and browse the map while predict and visualize the prediction result as an extra layer on original map. We introduced various type of data sources we use, like OpenStreetMap that is freely and publicly available or the Baidu Map's real-time traffic and POI data that require crawling, parsing and decoding raster image geospatial data. Next we discussed our general thoughts about feature engineering and stated three types of features used for prediction. We designed methods to reference external data sources about location popularity and uses Affinity Propagation clustering and A* routing algorithms while extracting non-local features. Then our model selection comparison and training, testing and evaluation process is introduced. We use multi-class linear SVM classifier. Finally, we go through the design and implementation details of our prediction system from the backend to the demo interface. We also show that our feature and model as well as the system is capable of predicting future traffic at a good accuracy with road network, points of interest and other spatial data, along with temporal and other related non-spatial data as input.

Keywords: Traffic prediction, Urban planning, Support vector machine (SVM), Affinity propagation, A* search algorithm, OpenStreetMap, Spatial-temporal data mining

Contents

References

Chapter 1 Introduction

1.1 Motivation and Goal

Traffic problems poses a great pressure on urban planning. Nowadays, as the number of vehicles in cities becoming larger and larger, traffic jams and accidents are more common and frequent. Thus, it is necessary that we look deep into the underlying cause and effect as well as simulation and prediction of the traffic flow under different situations. It can help urban planners to optimize roads and arrange land use for traffic, and predict the traffic in certain future time for users to better arrange their itinerary and routes.

The goal of this project is to develop algorithms, framework and a system to predict traffic conditions on any roads given a map formatted in OpenStreetMap data. The inputs to the problem include, the topological map data, time of the day, day of the week, location and the weather condition, etc. The output will be classified into 4 class labels: green (good), yellow (slow), red (congested) and deep red (extremely congested). This project will produce a web demo which is efficient and user friendly. We shall dig into various types of machine learning and data mining models, as well as simulation methods, mostly focusing on spatial-temporal data. By comparing those methods and fine-tuning the parameters, we would conclude the combinations that our system uses and reveal insights on the principle of how population move in urban areas. Also, as an engineering project, a system and a demo with interactive functionality that may be able to be put into industry is crucial.

1.2 Related Work

There are much work done by other researchers and industries because it is a real world issue that demand solution and will have great impact and contribution to predict traffic for people better traveling and relieve the heavy traffic by distributing people smartly. In industry, domestically, there are large tech companies like Baidu Map, which already have the functionality to predict traffic status of major roads at upcoming time, but few services for predicting and evaluating traffic given a completely new urban planning map exists. Many papers from Microsoft Institutes as well as MIT, have posed similar problem while proposing approaches like simulating by implementing network method, machine learning and even deep learning methods for prediction, which is a promising way for our problem too. Research work are done to tackle this problem from many different aspects. Çolak et al. (2016) and Simini et al. (2012) investigated and researched the underlying mobility pattern and understand the reasons of congested travel in urban areas. Schnitzler et al. (2014); Qi and Ishak (2014) proposed the usage of Hidden Markov Model as well as Gaussian Process in traffic prediction. Ren et al. (2014) predicts commuter flows in spatial networks using a radiation model based on temporal ranges. Xu et al. (2015,?) used spatial-temporal traffic prediction method. Pan et al. (2012); Mchugh (2015) utilize big data or real world transportation data for traffic prediction and visualization. Xu et al. (2014) proposed accurate and interpretable bayesian MARS for traffic flow prediction. Horvitz et al. (2012) studied methods, designs, and study of a deployed traffic forecasting service, which is a system similar to ours except they use self-deployed data source.

1.3 About OpenStreetMap

This is the main data source of our map data, apart from the historical traffic data from Baidu Maps. OpenStreetMap Coast (2004) is built by a community of mappers that contribute and maintain data about roads, trails, cafés, railway stations, and much more, all over the world. OpenStreetMap emphasizes local knowledge. Contributors use aerial imagery, GPS devices, and low-tech field maps to verify that OSM is accurate and up to date. OpenStreetMap's community is diverse, passionate, and growing every day. Our contributors include enthusiast mappers, GIS professionals, engineers running the OSM servers, humanitarians mapping disaster-affected areas, and many more. Also, the most important consideration of choosing OpenStreetMap as a data source is because it is open data: one are free to use it for any purpose as long as one credits OpenStreetMap and its contributors. If you alter or build upon the data in certain ways, you may distribute the result only under the same license, licensed under the Open Data Commons

Open Database License (ODbL) by the OpenStreetMap Foundation (OSMF) while The cartography in our map tiles, and our documentation, are licensed under the Creative Commons Attribution-ShareAlike 2.0 license (CC BY-SA).

1.4 Thesis Organization

In this thesis, we introduced a system that could be predicting future road traffic situations under given urban planning road network and point of interests as well as the given future time point etc. In chapter 2 the method and sources of data collection and crawling are introduced. In chapter 3, we mainly talk about the feature engineering and experimenting with different characteristics of the geospatial data. In chapter 4, we briefly introduce the machine learning and data mining model we used in the system, as well as the training and testing process, followed by the test result evaluations. Comparison with the baseline provided by Baidu Maps'prediction is presented. Next, in chapter 5, the implementation of our system is thoroughly introduced, with emphasis on the architecture along with the most important prediction engine, and the demo is shown. Finally, in chapter 6 we come to a conclusion of this project.

Chapter 2 Data Collection

2.1 Raw Map Data – from OSM

Data is essential while building the machine learning model and engineering the system. Our raw map data are all downloaded directly from the open-source OpenStreetMap daily data dump, available at `http://planet.osm.org/` as both a full dataset and a weekly change-set of the planet, showing all the updates that have been made to the planet by contributors. Since in the current stage of the project, all we are interested only in a small area of Eastern China containing Shanghai Municipality, we have used the `http://extract.bbbike.org/`, with the same data via the OpenStreetMap API for downloading our raw map data, where the customizable bounding box selection data export is available. The imported bounding box area starts from the south-west corner of approximately longitude 120.027°, latitude 30.551° to the north-east corner of approximately longitude 122.217°, latitude 31.805°.

The data format in the example data file Figure 2.2 is named as OSM XML format, a format for storing vector map road raw data. First of all, it is XML, XML is a so called meta format to provide even human readable data interexchange formats. Various file formats use this data tree structure to embed their data. The major tools in the OSM universe use an XML format following a XML schema definition that was first used by the API only. Basically it is a list of instances of our data primitives (nodes, ways, and relations). The contents of the file format include the following several parts:

- an XML suffix introducing the UTF-8 character encoding for the file
- an OSM element, containing the version of the API (and thus the features used) and the generator that distilled this file (e.g. an editor tool)
- a block of nodes containing especially the location in the WGS84 reference system
- the tags of each node
- a block of ways

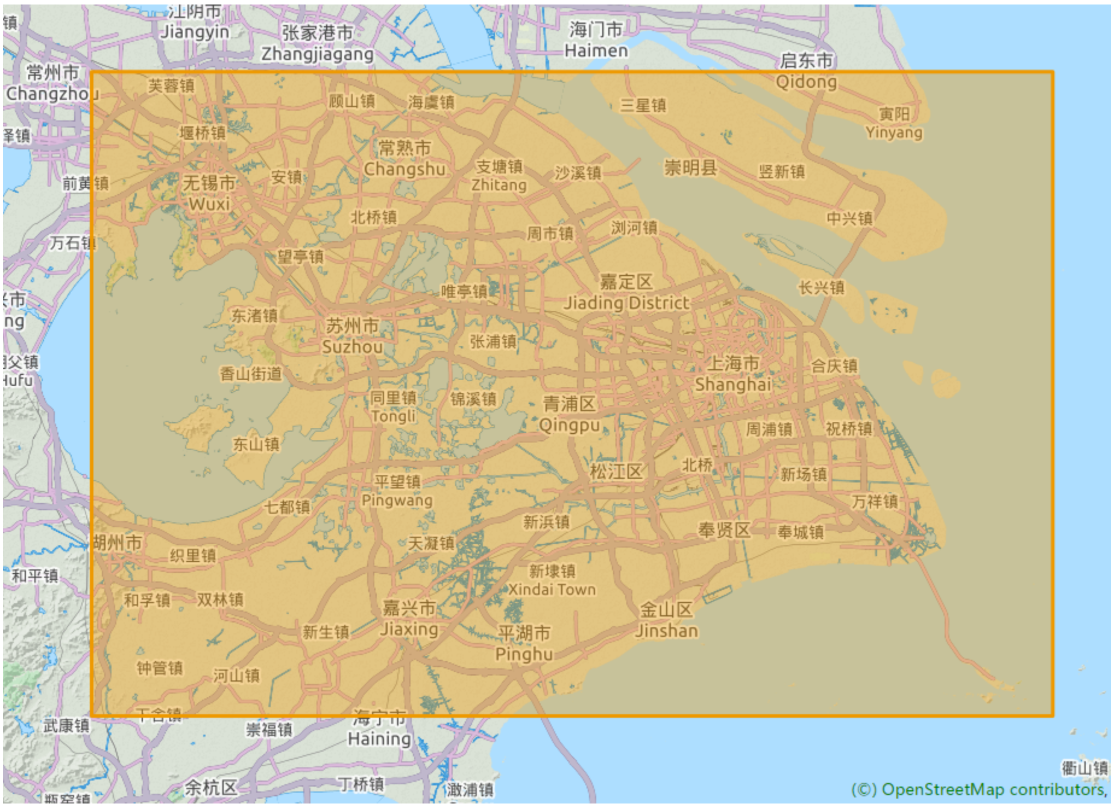

Figure 2.1 The bounding box area of the OSM data dump export

- the references to its nodes for each way
- the tags of each way
- a block of relations
- the references to its members for each relation
- the tags of each relation

Elements are the basic components of OpenStreetMap's conceptual data model of the physical world, also called the data primitives. They consist of nodes (defining points in space), ways (defining linear features and area boundaries), and relations (which are sometimes used to explain how other elements work together). All of the above can have one of more associated tags (which describe the meaning of a particular element). A node represents a specific point on the earth's surface defined by its latitude and longitude. Each node comprises at least an id number and a pair of coordinates. Nodes can be used to define standalone point features. For example, a node could represent a

```
<?xml version='1.0' encoding='UTF-8'?>
<osm version="0.6" generator="osmconvert 0.8.3">
  <bounds minlat="42.4543" minlon="-2.4761999" maxlat="42.4
  <node id="26861066" lat="42.471111" lon="-2.454722" versi
    <tag k="name" v="Camping La Playa"/>
    <tag k="tourism" v="camp_site"/>
    <tag k="operator" v="private"/>
    <tag k="addr:city" v="Logroño"/>
    <tag k="created_by" v="JOSM"/>
    <tag k="addr:street" v="Avda. de la Playa"/>
    <tag k="contact:fax" v="+34 941258661"/>
    <tag k="addr:country" v="ES"/>
    <tag k="addr:postcode" v="26006"/>
    <tag k="contact:email" v="info@campinglaplaya.com"/>
    <tag k="contact:phone" v="+34 941252253"/>
    <tag k="contact:website" v="http://www.campinglaplaya.c
    <tag k="internet_access" v="wlan"/>
    <tag k="addr:housenumber" v="6-8"/>
  </node>
  <node id="34793287" lat="42.4713587" lon="-2.4510783" ver
    <tag k="created_by" v="JOSM"/>
  </node>
  <node id="34793294" lat="42.4610836" lon="-2.4303622" ver
  <node id="34793297" lat="42.4548363" lon="-2.4287657" ver
  <node id="34793299" lat="42.4711478" lon="-2.4514125" ver
  <node id="34793300" lat="42.4590693" lon="-2.427438" vers
    <tag k="highway" v="crossing"/>
  </node>
```


Figure 2.2 The OpenStreetMap OSM data format example

park bench or a water well. Nodes are also used to define the shape of a way. When used as points along ways, nodes usually have no tags, though some of them could. For example, highway=traffic_signals marks traffic signals on a road, and power=tower represents a pylon along an electric power line. We mainly use nodes individually and separately to represent point features, like POIs (points of interests). A node can be included as member of relation. The relation also may indicate the member's role: that is, the node's function in this particular set of related data elements.

A way is an ordered list of between 2 and 2,000 nodes that define a polyline. Ways are used to represent linear features such as rivers and roads. Ways can also represent the boundaries of areas (solid polygons) such as buildings or forests. In this case, the way's first and last node will be the same. This is called a "closed way". Note that closed ways occasionally represent loops, such as roundabouts on highways, rather than solid areas. The way's tags must be examined to discover which it is. What we mainly focus on and utilize are the ways with specific tag highway=*, where The highway tag is the main tag used for identifying any kind of road, street or path. The highway type helps indicate the importance of the highway within the road network as a whole. Areas with holes, or with boundaries of more than 2,000 nodes, cannot be represented by a single way. Instead, the feature will require a more complex multi-polygon relation data structure.

A relation is a multi-purpose data structure that documents a relationship between two or more data elements (nodes, ways, and/or other relations). Examples include: A route relation, which lists the ways that form a major (numbered) highway, a cycle route, or a bus route. A turn restriction that says you can't turn from one way into another way. A multi-polygon that describes an area (whose boundary is the 'outer way') with holes (the 'inner ways'). Thus, relations can have different meanings. The relation's meaning is defined by its tags. Typically, the relation will have a 'type' tag. The relation's other tags need to be interpreted in light of the type tag. The relation is primarily an ordered list of nodes, ways, or other relations. These objects are known as the relation's members. Each element can optionally have a role within the relation. For example, a turn restriction would have members with "from" and "to" roles, describing the particular turn that is forbidden. A single element such as a particular way may appear multiple times in a relation.

Besides the basic nodes and ways, our information also requires from another important part of the data format, other than just longitude and latitude, and that is the tags of those basic elements. All types of data element (nodes, ways and relations) can have tags. Tags describe the meaning of the particular element to which they are attached. A tag consists of two free format text fields; a 'key' and a 'value'. Each of these are Unicode strings of up to 255 characters. For example, highway=residential defines the way as a road whose main function is to give access to people's homes. There is no fixed dictionary of tags, but there are many conventions documented on the OpenStreetMap wiki. If there is more than one way to tag a given feature, it's probably best to use the most common approach.

2.2 Traffic Data – from Baidu Map

The second part of the raw data collection is clearly the traffic situations, both currently and historically, on all the roads in the road network that we have previously downloaded and constructed from OpenStreetMap data. It is main drawback that such open source mapping service like OpenStreetMap does not provide such data, mainly because that the data source of traffic data is often acquired from taxi or drive GPSs as well as the authorities'traffic monitoring. So we need a third party data provider with most reliable traffic data. As the main study area is Shanghai now, after investigating various real-time traffic information provider, including Google Maps, AutoNavi and Bing Maps, etc., we came to the decision of using Baidu Maps as the source, because it is the biggest map provider and application with most users. In Figure 2.3, the traffic situation information is shown in the form of the green line as clear, yellow line as slow, red as congested and deep red as extremely congested. After digging and checking the network requests in the Baidu Maps webpage, there are two APIs in Baidu Maps that are useful for our purpose of getting traffic information data.

- The first API is in the form of `http://online1.map.bdimg.com/onlinelabel/?qt=tile&x=13204&y=3550&z=16`, see Figure 2.4 for response.

Figure 2.3 Traffic information on Baidu Map

Figure 2.4 A 256*256 px tile from Baidu Map

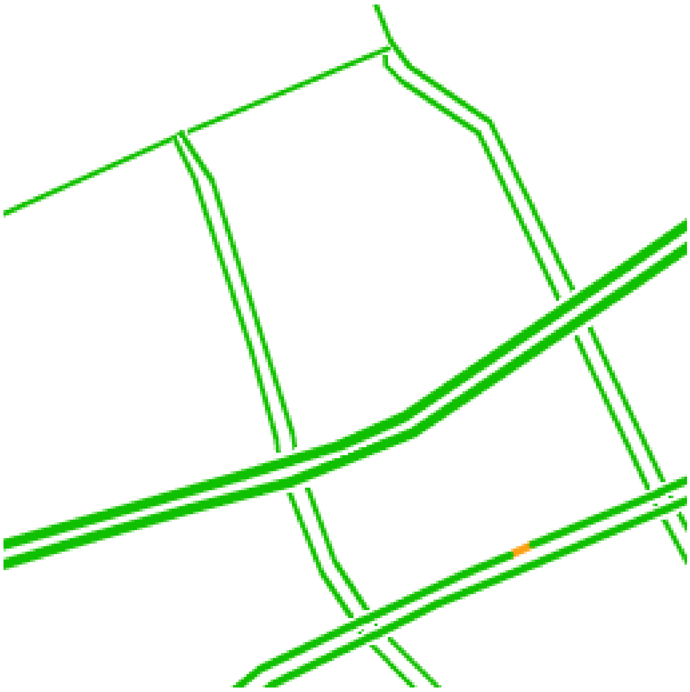

Figure 2.5 A traffic tile from Baidu Map, at the same place in Figure 2.4

- The second API is in the form of `http://its.map.baidu.com:8002/traffic/TrafficTileService?level=16&x=13204&y=3550&time=1464020567811`, see Figure 2.5 for response.

As you can see, the first API retrieves the tile image of the Baidu Map at given place (defined as tile coordinate x and y) and given zoom z. The second API is the traffic tile image overlay API that provides a PNG image with just polylines with color representing the traffic situation, with given x, y, z, along with a time parameter. We have a crawler that continuously download those data of the whole Shanghai urban area, getting tiles at zoom level 18 and composes all of those small tiles and traffic overlay into a large image. Because all the retrieve data are in raster image format, the next step would be parsing them using some image processing libraries.

We first get the ways from our previously dumped OpenStreetMap data, each way in the OSM data contains a series of nodes, with longitude and latitude, and those nodes connected following certain sequence number forms the roads and ways in the map. We interpolate some points between consecutive nodes connected, and get the coordinates of those in WGS-84 format. A simple API is also provided by Baidu to convert WGS-84 (the international standard) coordinates to BD-09 (a proprietary coordinated system used exclusively at Baidu Maps). After that conversion, we could transform the longitude and latitude to the Point Coordinates (x, y given z = 18) of Baidu Maps, and then use simple formula to calculate the exact pixel coordinates in the downloaded traffic tile images. Then we use naïve algorithm that find pixels within a distance threshold in terms of pixels from the sample point calculated above. If the point node belongs to a way that is one way tagged, then the search ends after only one color line found within threshold. Otherwise, two colored lines in most proximity to the point in direction orthogonal to the direction of the way are searched and found within threshold. Assigning those color line pixel matching results to the sample line would finish the task of parsing the raster image format of traffic data, and save them accordingly for later training use. See Figure 2.6 for an example processing result of the raster image traffic data. Some black dots that are not on the road are some other type of way nodes, like subway line which we are not taking care of.

2.3 Points of Interest – from Baidu Map

Though as an open data map service, OpenStreetMap provides some points of interest contributed by volunteers, but in mainland China, as the service is not popular and little know, the POIs are so sparse and scarce that nearly only a few POIs could be found in the most populated area of Shanghai. Because of this situation, we again turned to Baidu Maps for help, as it generously provides developers with a place query API that enable us to query places with given category tag of the POI within certain area defined by longitude and latitude bounding box. We used the following method to query all the POIs of all types and categories, both level 1 and level 2, in a recursive manner: first query the bounding box of all Shanghai area, and if the returned result is equal the API's maximum result limit, then split the bounding box into four, cutting half on each dimension adding small overlapping margin, until returned places lowered below the limit. After downloading all the POI data using Baidu Maps API, we processed the data with a script, removing the duplicates and then load all the data into our database, each place as a node mentioned earlier, with tags source=baidu and poitype=* and longitude

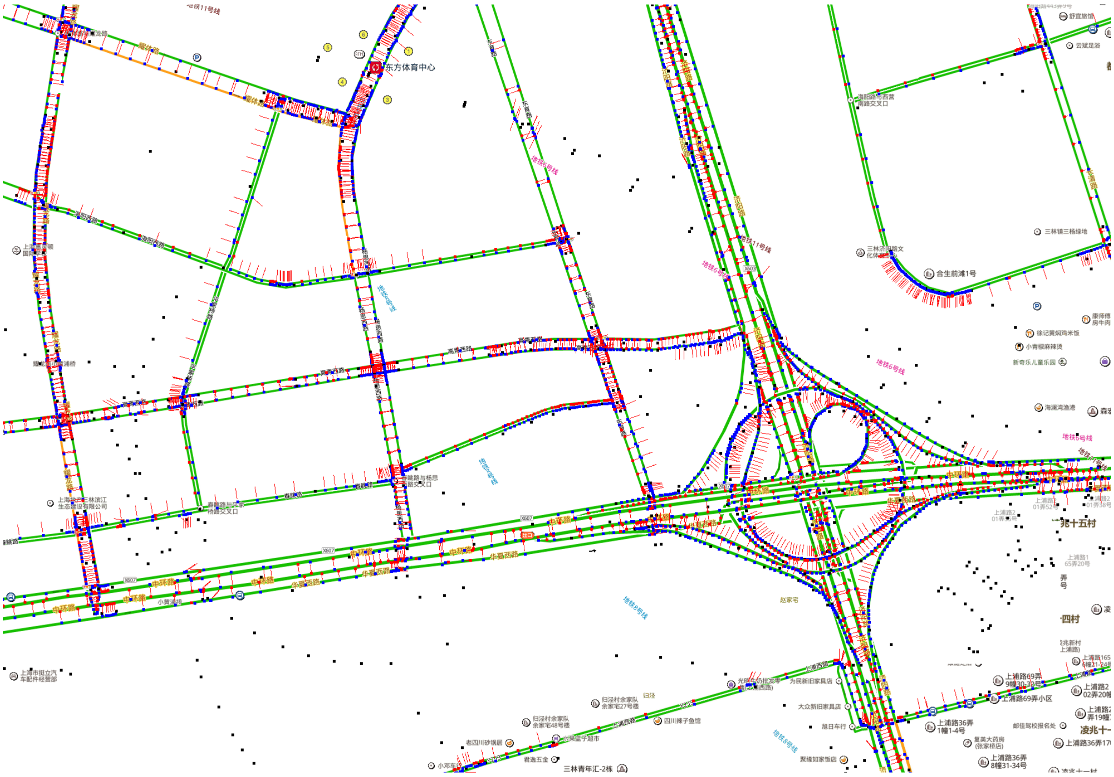

Figure 2.6 An example traffic parsing process of part of the area. The background is the image crawled and merged from several traffic tiles from Baidu Map. Black dots are all the nodes from our OpenStreetMap database mapped onto the location of Baidu Map. The red and blue dots are the parsing result of the image searching for each node's nearby color line pixel representing traffic information, and two of them in a pair denoting they are two directions of a two-way road

and latitude. Note that because the coordinate system of Baidu Maps and the real world coordinates used by OpenStreeMap differs, a conversion is required from BD-09 to WGS-84, but with an error of about 0.5 meters.

Chapter 3 Feature Extraction

3.1 Overview of the Machine Learning

Before going deep into any details of features, we shall introduce some basic ideas on the type of the problem and solution, machine learning. The project we are currently talking about is in fact a machine learning project, or say data mining project, which is connected in many ways. The problem itself can be abstracted be treated using various machine learning algorithms.

Machine learning is a subfield of computer science that evolved from the study of pattern recognition and computational learning theory in artificial intelligence. In 1959, Arthur Samuel defined machine learning as a "Field of study that gives computers the ability to learn without being explicitly programmed". Machine learning explores the study and construction of algorithms that can learn from and make predictions on data. Thy actually operate by building a model from example inputs in order to make data-driven predictions or decisions expressed as outputs, rather than following strictly static program instructions. Machine learning is closely related to and often overlaps with computational statistics; a discipline which also focuses in prediction-making through the use of computers. It has strong ties to mathematical optimization, which delivers methods, theory and application domains to the field. Machine learning is used widely in computing tasks where finding explicit algorithms is not feasible. Machine learning is sometimes conflated with data mining, where the latter sub-field focuses more on exploratory data analysis and is known as unsupervised learning. To summarize, in machine learning, we are more focused on the problem of optimizing the goal, while not required to find out the cause and effect behind. It is more like a black box process, in which the induction of the problem is done without investigating deduction. That is the main difference in my opinion compared to traditional artificial intelligence.

On the other hand, however, our problem can also be defined as a data mining problem, since our data is quite vast and large amount, but the underlying information is really not revealed completely. Data mining, as seen in its name, have a few outstanding characteristics:

- Automated prediction of trends and behaviors. Data mining automates the process of finding predictive information in large databases. Questions that traditionally required extensive hands-on analysis can now be answered directly from the data quickly.
- Automated discovery of previously unknown patterns. Data mining tools sweep through databases and identify previously hidden patterns in one step.

In our project, we are both discovering the cause and effect in the context of urban planning and mapping with the traffic density consequences, along with the task of predicting the future using historical data. Data mining techniques can yield the benefits of automation on existing software and hardware platforms, and can be implemented on new systems as existing platforms are upgraded and new products developed. When data mining tools are implemented on high performance parallel processing systems, they can analyze massive databases in minutes. Faster processing means that users can automatically experiment with more models to understand complex data. High speed makes it practical for users to analyze huge quantities of data. Larger databases, in turn, yield improved predictions. So the data collection progress described in previous chapter is still up and running, crawling the data every 30 minutes. Only with more data and more features getting engineered, the better result we could have. It is actually the dimension of the database that matters.

We could use an example on database to say that features and data instances can be larger in both depth and breadth:

- More columns. We must often limit the number of variables they examine when doing hands-on analysis due to time constraints. Yet variables that are discarded because they seem unimportant may carry information about unknown patterns. High performance data mining allows users to explore the full depth of a database, without preselecting a subset of variables.

- More rows. Larger samples yield lower estimation errors and variance, and allow users to make inferences about small but important segments of a population.

3.2 Introduction to Feature Engineering

Having been familiar with the history and differences between the subfields in the whole machine learning field, we can to some extent categorize this project on predicting the traffic situation at any future time given any map compositions into a machine learning problem categories because we have the process of training and modeling with human interference and knowledge, which is basically classification problem in supervised learning, while also seeking to predict and find patterns by raw data amounts and computing power. So in this chapter we shall mainly introduce the methods and goals for feature extracting and feature engineering, which is the most important thing a researcher could do when tackling real world problems with existing model. That is applicable rule in most of the cases since features you extracted contributes largely to the final prediction result.

To make an input data set for later training and testing, we need to dig deep into the collected data, both in massive amount and high dimensions. This is called feature extraction and we could not just feed all the raw data into the model training because that would be impossible to grab the potential pattern and cause and effect, also impossible for the computing power we currently have. Our data are like a huge graph with nodes and connected edges, while edges and nodes have their tags and weights. We need a process that could extract and output the features that is useful for this particular goal, predicting the traffic in the future and in case of mapping changes. Only after that we could feed those much smaller and denser data into training process. In machine learning, feature extraction starts from an initial set of measured data and builds derived values (features) intended to be informative and non-redundant, facilitating the subsequent learning and generalization steps, and in some cases leading to better human interpretations. Feature extraction is related to dimensionality reduction. When the input data to an algorithm is too large to be processed and it is suspected to be redundant, then it can be transformed into a reduced set of features (also named a features vector).

This process is called feature selection. The selected features are expected to contain the relevant information from the input data, so that the desired task can be performed by using this reduced representation instead of the complete initial data. Feature extraction involves reducing the amount of resources required to describe a large set of data. When performing analysis of complex data one of the major problems stems from the number of variables involved. Analysis with numerous variables generally requires a large amount of memory and computation power, also a classification algorithm sometimes causes a common problem that it tend to over-fit to training samples, so that generalize poorly to new samples. Feature extraction is a general term for methods of constructing combinations of the data variables to avoid these problems while still describing the data with sufficient accuracy. The best results are often achieved when an expert constructs a set of application-dependent features, a process called feature engineering.

Feature engineering is the process of using domain knowledge of the data to create features that make machine learning algorithms work and it is fundamental to the application of machine learning, and is both difficult and expensive. Manual feature engineering's necessity can be replaced by automated feature learning. Feature engineering is an informal topic, but it is considered essential in applied machine learning. Actually like Andrew Ng said in his Machine Learning and AI via Brain simulations, coming up with features is difficult, time-consuming, requires expert knowledge. "Applied machine learning" is basically feature engineering. We have exact feeling when doing our job, and actually that is why there is a chapter specified for this. As a human being, we have some roles when applying machine learning. Machine learning provides you with extremely powerful tools for decision making but until a breakthrough in AI, the role of the developer's decision will still be crucial. Our responsibility includes setting up the correct problem to be optimized (it's far from straightforward in the real world), choosing a model, choosing a learning algorithm (or a family of algorithms), finding relevant data, designing features, feature representation, feature selection, etc. Two components in feature engineering is that: first, understanding the properties of the task you're trying to solve and how they might interact with the strengths and limitations of the model you are going to use, and second, experimental work where you test your expectations and find out what actually works and what doesn't.

3.3 Local Geospatial Features

While digging on the information we have and brainstorming to achieve a list of features that may become useful, the first set of features that comes into our mind is the information that could best describe the local situation of a given place or point. For example, the road network density in proximity of the node. If the road is along with a lot of crossings with other roads in a relatively small length, it means that there may exists many traffic lights and will inevitably slow the traffic down. The denser network would also imply that many cars will tend to turn right or left other than just going straight. As what we are always trying to do is finding out what may affect the traffic behavior in such a small area, the local features are really important and worth standing out. So we calculated the average distance between the road crossing nodes in both directions from the node. It is trivial by using a formula to calculate the earth distance given two nodes' coordinates.

Besides the road network density described as the average distance of each road crossing with others, both forward and backward direction, we would also take POIs into great account. If you think the people living in a city's behavior thouroghly, it is actually the points of interests like buildings for work and shopping mall for shopping as well as restaurants for dining that matters the most and affects when and where the people would like to drive to. So the local features on what is besides or near the node on the way is significant. People do go to a specific place or road for some good reasons! From the Baidu Maps points of interest data which we have crawled before, we have manually re-categorized the Baidu's POI types into our own, a total of 37 types. We did investigate the typing hierarchy that Baidu provides and since it is a 2 level tree format, we merged some of the types and also deleted some of the less important ones regarding the behavior of people who drove cars. Then we select all the second level types as ours. In Table 3.1 there is the original Baidu Maps category tree, and refer to Table 3.2 for more details on our redesigned POI types. All the categories in the second level column is taken.

Table 3.1 Original Baidu Maps POI category hierarchy

First Level Categories	Second Level Catagories
美食	中餐厅、外国餐厅、小吃快餐店、蛋糕甜品店、咖啡厅、茶座、酒吧
酒店	星级酒店、快捷酒店、公寓式酒店
购物	购物中心、超市、便利店、家居建材、家电数码、商铺、集市
生活服务	通讯营业厅、邮局、物流公司、售票处、洗衣店、图文快印店、照相馆、房产中介机构、公用事业、维修点、家政服务、殡葬服务、彩票销售点、宠物服务、报刊亭、公共厕所
丽人	美容、美发、美甲、美体
旅游景点	公园、动物园、植物园、游乐园、博物馆、水族馆、海滨浴场、文物古迹、教堂、风景区
休闲娱乐	度假村、农家院、电影院、KTV、剧院、歌舞厅、网吧、游戏场所、洗浴按摩、休闲广场
运动健身	体育场馆、极限运动场所、健身中心
教育培训	高等院校、中学、小学、幼儿园、成人教育、亲子教育、特殊教育学校、留学中介机构、科研机构、培训机构、图书馆、科技馆
文化传媒	新闻出版、广播电视、艺术团体、美术馆、展览馆、文化宫
医疗	综合医院、专科医院、诊所、药店、体检机构、疗养院、急救中心、疾控中心
汽车服务	汽车销售、汽车维修、汽车美容、汽车配件、汽车租赁、汽车检测场
交通设施	飞机场、火车站、地铁站、长途汽车站、公交车站、港口、停车场、加油加气站、服务区、收费站、桥
金融	银行、ATM、信用社、投资理财、典当行

房地产	写字楼、住宅区、宿舍
公司企业	公司、园区、农林园艺、厂矿
政府机构	中央机构、各级政府、行政单位、公检法机构、涉外机构、党派团体、福利机构、政治教育机构

Table 3.2 Our redesigned POI category hierachy, second level types taken

First Level Categories	Second Level Catagories
交通设施	停车场、公交车站、加油加气站、地铁站、收费站、服务区、桥、港口、火车站、长途汽车站、飞机场
休闲娱乐	休闲娱乐
公司企业	公司、农林园艺、厂矿、园区
医疗	医疗
房地产	住宅区、写字楼、宿舍
政府机构	政府机构
教育培训	高等院校、其他
文化传媒	文化传媒
旅游景点	海滨浴场、游乐园、风景区、其他
美食	美食
购物	小（便利店，商铺）、大（其他）
运动健身	体育场馆、健身中心、极限运动场所
酒店	酒店
金融	金融服务
其他	其他

The features we designed for POIs are in combination of the nearby road crossings related to the previously discussed average distance. In this set of features, we proposed that the POI density of both the current node and the forward and backward nodes located in the next or last road crossing shall have a role. We counted the number of each type of the points of interest around 200 meters away within the node. Such density counting with the same radius parameter is also done for the next 1, 2 and 3 road crossing nodes in both forward and backward directions, as well as the next crossing nodes in the road network both left and right to the direction of the current node's segment. So in total there is 37 types of points of interest for each node, and one given node have 3 level forward crossing point, 3 level backward crossing point, 1 left and 1 right direction crossing nodes, there are $37 * 8 = 296$ features being related and attached for each node. It takes a vast majority of our extracted feature list, and by such combination of minor features, there is already cause redundancy, but we considered its importance of preserving information in an obvious way crucial and we can afford to calculate.

3.4 Clustering and Routing – Non-Local Geospatial Features

As stated in the above section, the local features already play a large part of the roles in the features. However, not all traffic problems and behaviors are only related to the locality of the place where all the cars and traffic goes through. Actually we need to figure out a way to describe and better understand the driver's motivation and habits. People do not just drive around the city finding the shops or the supermarkets on the sides of the road, they drive every day to work and back home. These commuters are making up a large portion the traffic jam that happens every day on rush hours like in the morning or in the late afternoon on elevated roads, highways, etc. Taking an example in Shanghai for example, many people lives in the suburban areas that is outside the outer ring road of the city. Most of the large companies that is really having large numbers of employees are most possibly located in the urban area inside the middle ring road. A long journey and distance for commuters is a great example showing that local geospatial data are not enough. It does not capture the potential information of where the people would most likely come from and where would they most likely. Temporal data also affecting the problem. In the weekdays, rush hours'drivers contribute to the most of the traffic jam situations, on the other hand, in the weekend, many people would stay in some shopping mall and going out to the city center to have a nice dinner. The peak travelling hours are varying and changing.

After proposing that local features are not sufficient, we need to figure out a quantitative set of features that best describe the driver's preference on roads. Some roads will get really crowded and slow because too many drivers are going to choose the route including the road. For example, every morning and late afternoon, the S4 Hujing Expressway and Humin Elevated Road are both under heavy traffic because there are many people living in the Minhang District of Shanghai, which is a large place for residential areas. The road is also the easiest road to drive that connects the urban area of Xujiahui District, which has all the commercial buildings and shopping malls. Examples like this exists in all aspects of urban life and are quite common. Actually most people on the road lives in the similar place and go to work or eat in the similar places too, and because of the clustering property of human society and urban planning, that gives one of the reasons why roads are crowded. That is we need to identify the "functional areas"in the city that has three types of functionality: shopping, residential and commercial.

In order to solve this, we need to first find out where people usually live, which is the residential area in the city; where people usually drive to work, which is the business area in the city; and where people usually dine outside and go shopping, which is the commercial area in the city. We need to find out those large clustered points of interest by using clustering algorithms which prevails in the data mining fields.

Clustering the scattered data is among one of the most important unsupervised learning problem. Its main goal is deals with finding a structure in a collection of unlabeled data, which could also be described as "the process of organizing objects into groups whose members are similar in some way". A cluster is therefore a collection of objects which are "similar"between them and are "dissimilar"to the objects belonging to other clusters. See Figure 3.1 for a graphical example:

We can easily see in the figure that easily the data can be divided into 4 clusters. The similarity criterion in this case is distance: two or more objects belong to the same cluster if they are "near" each other according to a given distance which is geometrical distance in this case, also known as distance-based clustering. The goal of clustering is to determine the intrinsic grouping in a set of unlabeled data. But in order to decide what constitutes a good clustering, we show that there exists no absolute best criterion independent of the final goal of the clustering. As a result, it is us who must supply this

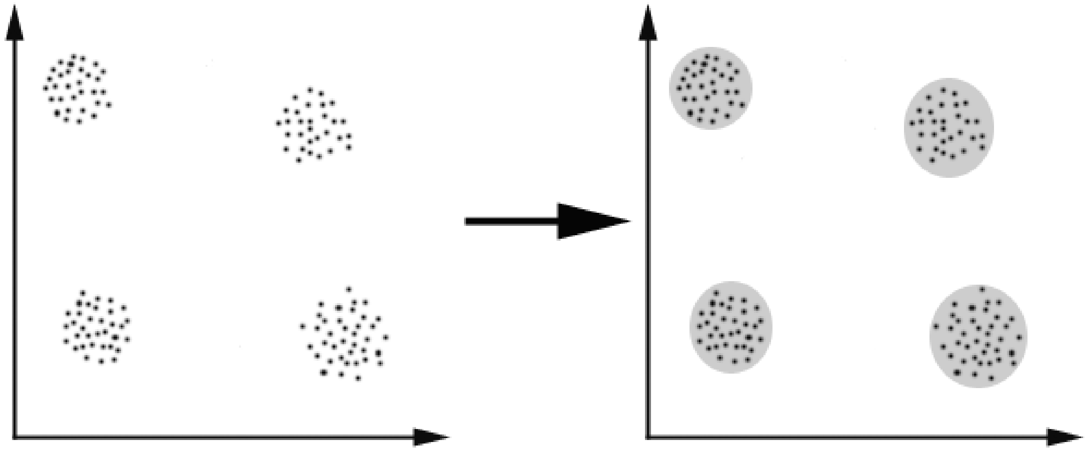

Figure 3.1 An example of clustering process

criterion that the result of the clustering will suit the needs.

3.4.1 Obtaining Data for Clustering

Before trying out a variety of the clustering algorithms we have currently, we need to collect the raw data first and then apply those algorithms to them. Currently in our database, we have the OpenStreetMap's geospatial data like road network maps, and the nodes (which we also call them points of interest) crawled, processed and loaded into the database. It seems that we already have solid knowledge base on the sheer number of POIs we have now. However, we lack the popularity of the points of interest information. We wanted to know how many people and how frequently people would go to the specific shopping mall or which building have large amounts of employees that work there. An approach is designed to solve this lack of knowledge problem by cross referencing and collaborate with crawled data of other websites. In our case, we used two external data source, Dianping.com and place.weibo.com, which is both very popular among Chinese people and have a large user base. We primarily used the Dianping.com for the extensive categorical points of interest information of the names, address, rating, price as well as the comments and number of comments of the point. We wrote a Python crawler script to crawl a whole set of data from Dianping.com, mainly

Figure 3.2 An example page of the POI listings of business sites in Shanghai on Dianping.com

consisting 3 types of the functional areas: shopping malls, residential areas and commerce buildings in Shanghai. See Figure 3.2 for an example page with information of the places.

After we obtained the places in Shanghai from Dianping.com, we can parse the information of each place obtained. A special program is written to parse the Chinese address shown in the webpage, and extracts only the road name and city name in it. It is actually an address parser using extensive regular expressions to match the address in format that could extract the name of road. The reason for this is to have a better link between the name and address, and better support fuzzy matching while later searching on Weibo Place. Normally there are many shops and restaurants in one shopping mall, and many companies in one business building, so with the help of address parsing, it is much more accurate to find all places associated to that large place. Place.weibo.com is a website for searching location based information that people attached with their status provided by Weibo, a popular social media network in China, that supports searching the place with keywords. Its result is useful for us to determine and quantize the popularity among the people of given place of interest that we obtained earlier in Dianping.com. As shown in Figure 3.3 the information of the search result of a specific place contains the number of posts containing the location, and how many people have come here before and the number of photos users have uploaded.

We search for each of the places in Weibo Place by place name that we obtained earlier in Dianping.com and filtered the search result with the road name parsed from the address to ensure that only places with the same address are being counted. The sum of those numbers of posts, photos, check-ins are saved to output file, which will be processed to be our data for clustering later. We sorted by the sum as popularity of the place inside the whole lists of those places, and get a top 1000 most popular list of places of business sites, residential areas and shopping malls. We have made the plots of these data as scattered data in geographical coordinates. See Figure 3.4, 3.5 and 3.6 for the distribution and density of each of the 3 functional areas. X-axis dimension is the latitude and Y-axis dimension is the longitude.

3.4.2 Clustering the Data Collected

Now that we have processed top 1000 most popular lists of three types of places, we can apply the clustering algorithm to find out the clustered functional areas that we wanted. Several of the famous and popular clustering algorithms exist, but the methods of clustering mainly fall into two categories: the first is hierarchical (agglomerative) that initially, each point is in cluster by itself, and then repeatedly combine the two “nearest”into one. The second is point assignment, that is it maintains a set of clusters, and place points into their “nearest”cluster. Many algorithms exist for dealing with such problems, for example the very famous algorithm called the k-means, as well its variant

Figure 3.3 An example page of the location name search results of Weibo on place.weibo.com

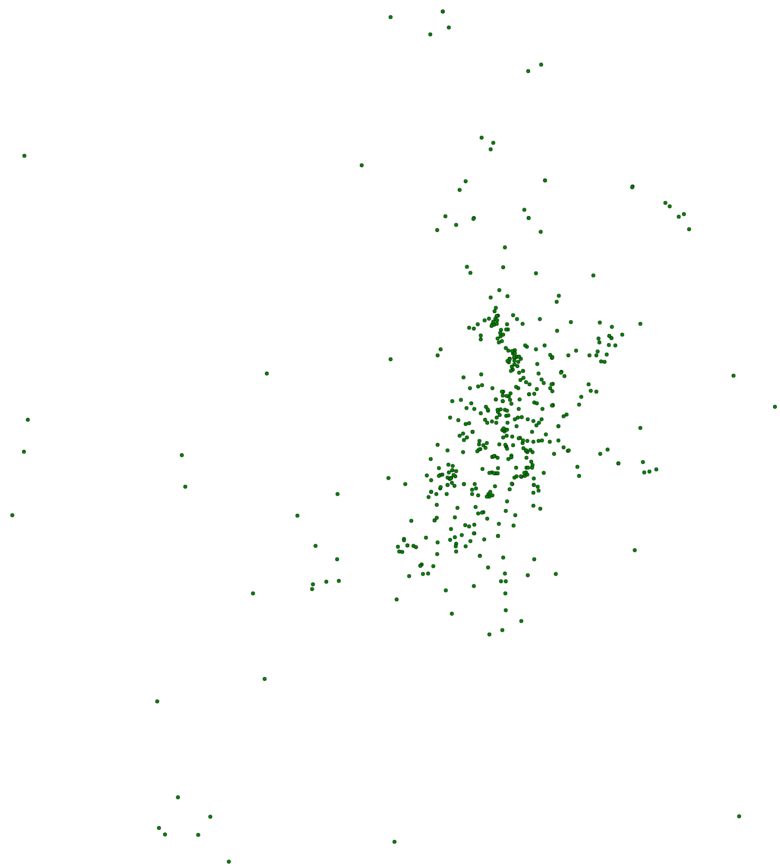

Figure 3.4   Scattered plot of top business site areas in Shanghai

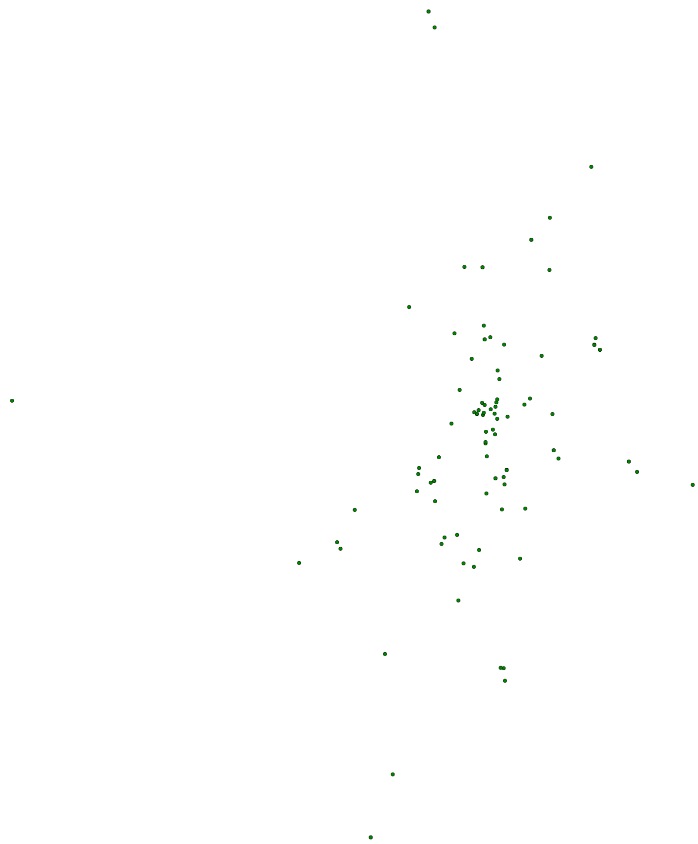

Figure 3.5 Scattered plot of top shopping mall areas in Shanghai

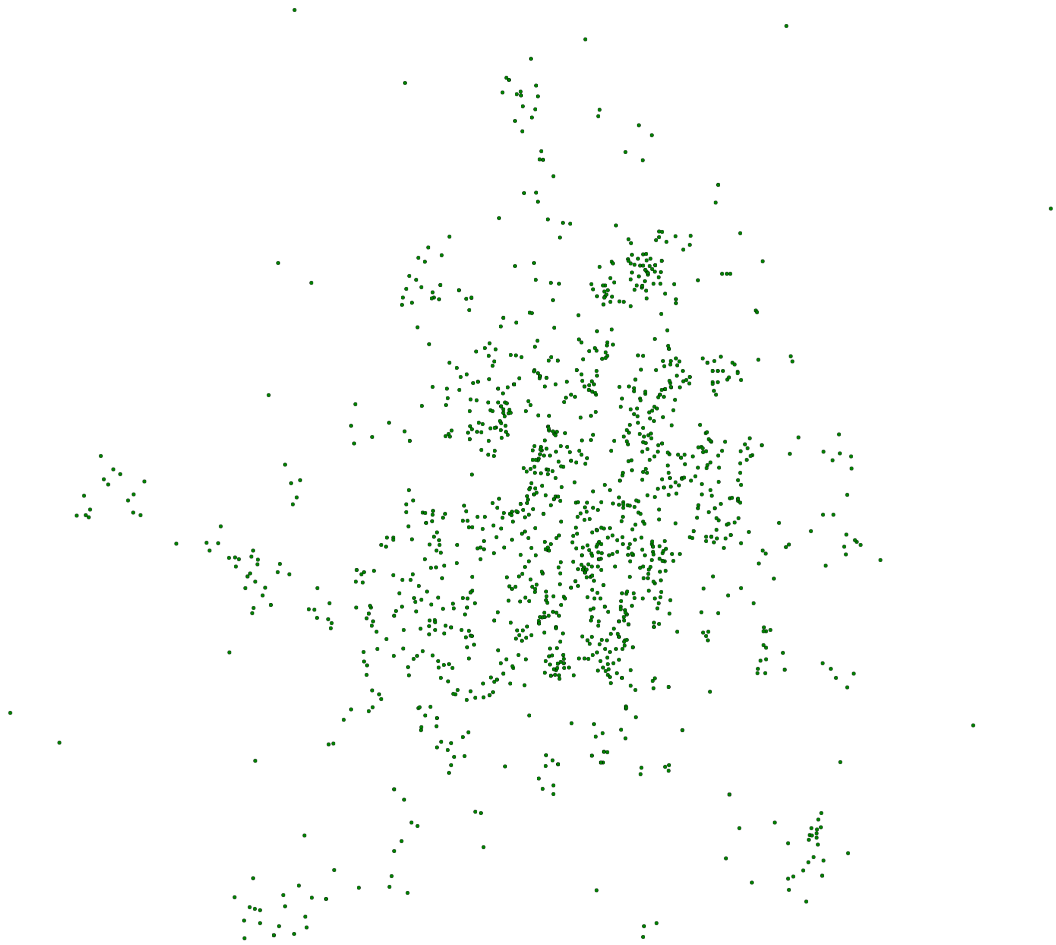

Figure 3.6 Scattered plot of top residential areas in Shanghai

BFR (Bradley-Fayyad-Reina) algorithms for large dataset and some other hierarchical clustering methods, DBSCAN, and Gaussian Mixtures, etc. In this case, however, after studying and experimenting with those algorithms, we selected to use Affinity Propagation algorithm Frey and Dueck (2007). It's said to be fast, efficient and do not need to specify the number of cluster beforehand, which suits our case well since we want fully automatic clustering without knowing how many those different functional areas there are. It creates clusters by sending messages between pairs of samples until convergence. A dataset is then described using a small number of exemplars, which are identified as those most representative of other samples. The messages sent between pairs represent the suitability for one sample to be the exemplar of the other, which is updated in response to the values from other pairs. This updating happens iteratively until convergence, at which point the final exemplars are chosen, and hence the final clustering is given. Affinity Propagation can be interesting as it chooses the number of clusters based on the data provided. For this purpose, the two important parameters are the preference, which controls how many exemplars are used, and the damping factor. The main drawback of Affinity Propagation is its complexity. The algorithm has a time complexity of the order $O(N^2T)$, where N is the number of samples and T is the number of iterations until convergence. Further, the memory complexity is of the order $O(N^2)$ if a dense similarity matrix is used, but reducible if a sparse similarity matrix is used. This makes Affinity Propagation most appropriate for small to medium sized datasets.

Algorithm description: The messages sent between points belong to one of two categories. The first is the responsibility $r(i, k)$, which is the accumulated evidence that sample k should be the exemplar for sample i. The second is the availability $a(i, k)$ which is the accumulated evidence that sample i should choose sample k to be its exemplar, and considers the values for all other samples that k should be an exemplar. In this way, exemplars are chosen by samples if they are (1) similar enough to many samples and (2) chosen by many samples to be representative of themselves. More formally, the responsibility of a sample k to be the exemplar of sample i is given by:

$$r(i,k) \leftarrow s(i,k) - max[a(i,\acute{k}) + s(i,\acute{k}) \forall \acute{k} \neq k] \tag{3-1}$$

Where $s(i,k)$ is the similarity between samples i and k. The availability of sample k to be the exemplar of sample i is given by:

$$a(i,k) \leftarrow min[0, r(k,k) + \sum_{\acute{i}\ s.t.\ \acute{i} \notin \{i,k\}} r(\acute{i},k)] \tag{3-2}$$

To begin with, all values for r and a are set to zero, and the calculation of each iterates until convergence.

In engineering, we used the open source machine learning library called scikit-learn Pedregosa et al. (2011) which has the affinity propagation algorithm included. So we simply feed the data in with a small Python script and we have some quite amazing clustering result. For example, in the clustering of the data of top business sites obtained earlier, eventually it generates 23 clusters and the Silhouette Coefficient is 0.63. It means that it works relatively good. If the ground truth labels are not known, evaluation must be performed using the model itself. The Silhouette Coefficient is an example of such an evaluation, where a higher Silhouette Coefficient score relates to a model with better defined clusters. The Silhouette Coefficient is defined for each sample and is composed of two scores: a: The mean distance between a sample and all other points in the same class. b: The mean distance between a sample and all other points in the next nearest cluster. The Silhouette Coefficient s for a single sample is then given as:

$$s = \frac{b - a}{max(a,b)} \tag{3-3}$$

The Silhouette Coefficient for a set of samples is given as the mean of the Silhouette Coefficient for each sample. The score is bounded between -1 for incorrect clustering and +1 for highly dense clustering. Scores around zero indicate overlapping clusters. The score is higher when clusters are dense and well separated, which relates to a standard concept of a cluster. For clustering result, we can see Figure 3.7 for the clustered

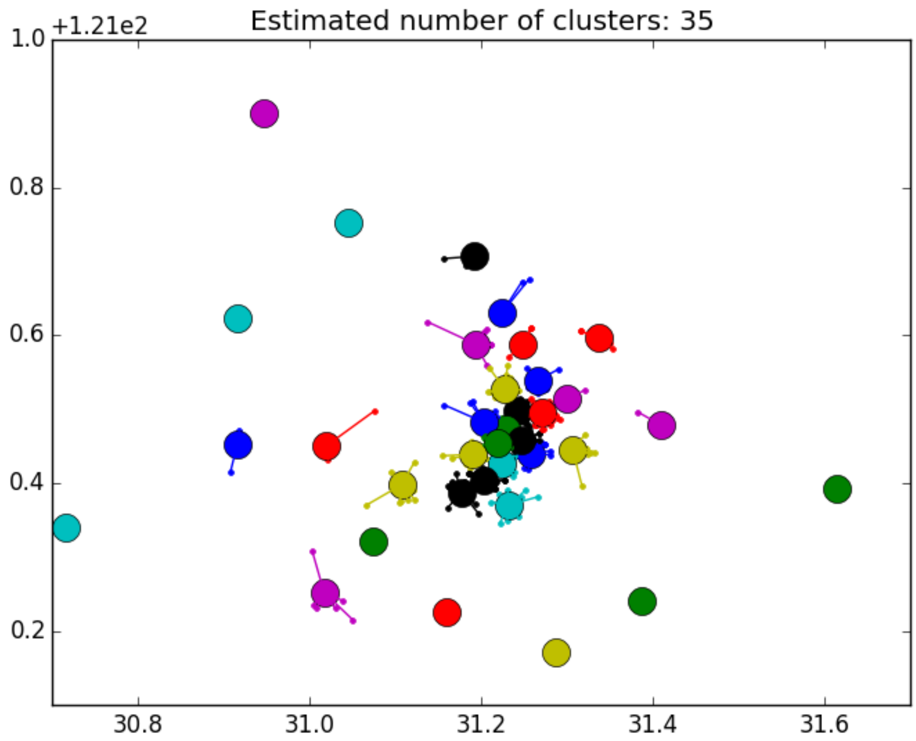

Figure 3.7 The clustering output by running top business site areas with Affinity Propagation algorithm, x-axis being Latitude and y-axis being Longitude

output's plot of the top business sites found. Those clusters are exactly the functional areas of this particular type that we discussed earlier and at first glance it seems good.

3.4.3 Routing between Functional Areas

In the previous chapter, we used the affinity propagation algorithm to cluster the top popular points of interest for each type of the three category. At the time we have three sets of the functional areas of the type business site, residential area and shopping area. We have the knowledge of where they are located at, since the cluster algorithm returns an exemplar for representing the whole nearby area. In order to reflect this information on the road network, we need to find out the connecting routes that people and commuters use for going to work and back home as well as to malls for eating and shopping. Imagine a scenario that a person who lives in Minhang District near Shanghai Jiao Tong

University, is working in a company located at Xujiahui. So his living area is actually within the functional area of residence, and Xujiahui is both a business site area and shopping mall area. For his weekdays, he would most probably take the S4 Expressway and into Humin Elevated Road to get to work, so these highways will be under heavy traffic during rush hours, making all the nodes and way segments on it a popular way of choice On Friday night after work for example, he would like to drive to the South Shanxi Road to have dinner in IAPM shopping mall and shop there with his girlfriend, so on such a weekend night, he would take Huaihai Road to move from the business site area to a shopping mall area, and Huaihai Road may become very crowded for this popularity, and so does the nodes on this road.

In order to find out what ways are popular or not because of the reason that these nodes and segment on this particular way are the connecting components that lies on the road among which is crucial in the most popular routing choice. As many people would use navigation software, finding the recommended routes between those discovered types of functional areas now proves to be important.

When talking about routing algorithms on a road network, we would instantly consider the map information being represented as a directional graph with weight on the edges. The edges would be the road segments connecting nodes. Our new problem of finding the most popular route between two locations now is being translated into the problem of finding the shortest path in a large graph between two vertices. To be more accurate, it should be the cheapest cost path because we do not just want the route that is short in mere distance, we would like to take road type and so on into consideration. So a weighted shortest path problem should best describe this, and the weight is affected by road type, for example the elevated roads or highway that have a better speed and priority should cost less weight than the secondary and ordinary roads with crossings and traffic lights making them more favorable. Also, one-way is also an important deciding factor in routing algorithms because it would simply disable or prohibit such edges being reached in certain direction. In our database there is always a tag with the way elements, showing the road hierarchy and one-way flag.

We consider a directed graph $G = (V, E)$ with n nodes and $m = (n)$ edges. An edge

(u, v) has the non-negative edge weight $w(u, v)$. A shortest-path query between a source node s and a target node t asks for the minimal weight $d(s, t)$ of any path from s to t. In static routing, the edge weights do not change so that it makes sense to perform some pre-computations, store their results, and use this information to accelerate the queries. Obviously, there is trade-off between query time, pre-processing time, and space for pre-processed information. In particular, for large road networks it would be prohibitive to pre-compute and store shortest paths between all pairs of nodes.

Dijkstra's Algorithm Dijkstra (1959) is the classical algorithm for route planning —maintains an array of tentative distances $D[u]{\geq}d(s, u)$ for each node. The algorithm visits (or settles) the nodes of the road network in the order of their distance to the source node and maintains the invariant that $D[u] = d(s, u)$ for visited nodes. We call the rank of node u in this order its Dijkstra rank $rks(u)$. When a node u is visited, its outgoing edges (u, v) are relaxed, i.e., $D[v]$ is set to $min(D[v], d(s, u) + w(u, v))$. Dijkstra's algorithm terminates when the target node is visited. The size of the search space is $O(n)$ and $n/2(nodes)$ on the average. We will assess the quality of route planning algorithms by looking at their speedup compared to Dijkstra's algorithm, i.e., how many times faster they can compute shortest-path distances.

The algorithm that we use in our routing purpose is A* Search algorithm Hart et al. (1968), also known as Geometric Goal Directed Search. The intuition behind goal directed search is that shortest paths 'should'lead in the general direction of the target. A* search achieves this by modifying the weight of edge (u, v) to $w(u, v){-}(u) + (v)$ where (v) is a lower bound on $d(v, t)$. Note that this manipulation shortens edges that lead towards the target. Since the added and subtracted vertex potentials(v) cancel along any path, this modification of edge weights preserves the shortest paths. Moreover, as long as all edge weights remain non-negative, Dijkstra's algorithm can still be used. The classical way to use A* for route planning in road maps estimates $d(v, t)$ based on the Euclidean distance between v and t and the average speed of the fastest road anywhere in the network. Since this is a very conservative estimation, the speedup for finding the quickest routes is rather small. Lastly there is heuristic that evolved during recent years. In the last decades, commercial navigation systems were developed which had to handle ever more detailed descriptions of road networks on rather low-powered processors. Vendors resolved to heuristics still used today that do not give any performance guarantees: use A□ search with estimates on $d(u,t)$ rather than lower bounds; do not look at 'unimportant'streets, unless you are close to the source or target. The latter heuristic needs careful hand tuning of road classifications to produce reasonable results but yields considerable speedups. Refer to the Listing 3.1 for a pseudocode of A* searching algorithm.

```
// OPEN list consists on nodes that have been visited but not expanded (meaning that sucessors have not been explored yet). This is the list of pending tasks.
// CLOSE list consists on nodes that have been visited and expanded (sucessors have been explored already and included in the open list, if this was the case).
initialize the open list
initialize the closed list
put the starting node on the open list (you can leave its f at zero)

while the open list is not empty
  find the node with the least f on the open list, call it "q"
  pop q off the open list
  generate q's 8 successors and set their parents to q
  for each successor
    if successor is the goal, stop the search
    successor.g = q.g + distance between successor and q
    successor.h = distance from goal to successor
    successor.f = successor.g + successor.h

    if a node with the same position as successor is in the OPEN list which has a lower f than successor
      then skip this successor
    if a node with the same position as successor is in the CLOSED list which has a lower f than successor
      then skip this successor
    otherwise
      add the node to the open list
```

```
  end
  push q on the closed list
end
```

Listing 3.1 A* searching algorithm in pseudocode

In the engineering process, we implemented A* algorithm to read the OpenStreetMap data format, along with two coordinates representing the starting point and ending point of the routing. We used the tags about the road that the ways in the database have to create a customized weight for different type of roads. We are routing each location in the clustered functional areas to those other locations in another type of functional areas. So we have 3 groups of routing being done: the routes between locations in residential areas and business sites areas, the routes between locations in residential areas and shopping mall areas, and the routes between locations in business sites areas and shopping mall areas. An example routing after running our program is plotted in Figure 3.8, note that the map plots all valid roads of the bounded area in Shanghai, and the red polylines which take a large majority is the roads that has not been searched. Those in blue are the roads that the A* algorithms runs through and calculated. The green one is the final optimal route found. We have all the node ID and way ID information so that we can identify easily.

In order to make the routing information we have all done into our feature list of each nodes, or road segments, we designed three features for each node to represent the hot or popular ways, which are the numbers of occurrences for this node to be in the all routes between those 3 clusters of functional areas. To be clear, there are the number of the best routes between residential areas and business site areas that passes just through this particular node, and the number between residential areas and shopping mall areas as well as the number between business sites area and shopping mall areas. In this way, we can have each node having the features that represent the level of popularity of itself as being chosen by people when thinking of driving from on cluster to another.

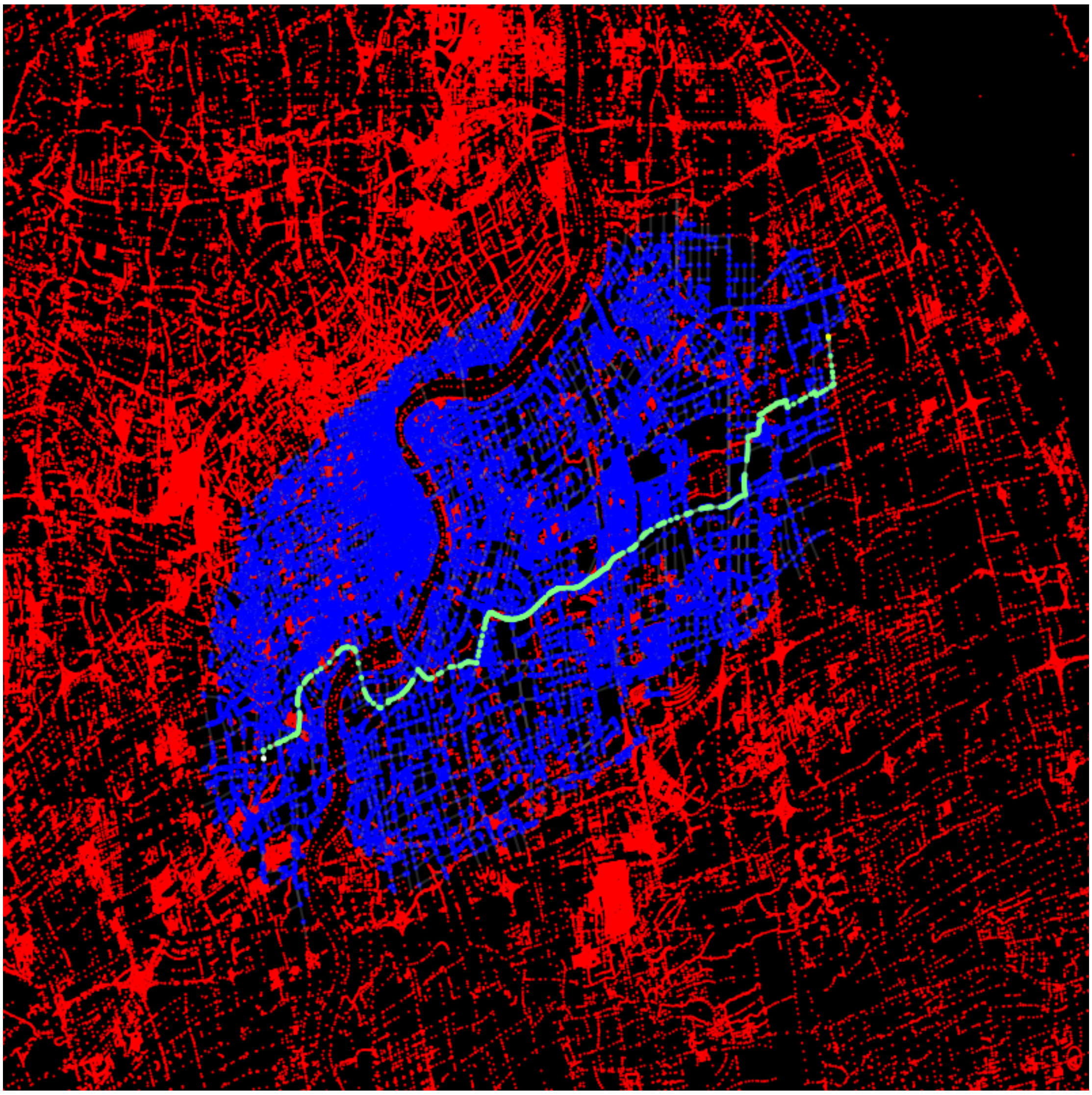

Figure 3.8 An example graphical output of the optimal routing program from one place to another

3.5 Non-Geospatial Implicit Features

Besides the features related to location information and the geographical properties around them that we discussed about in the previous sections, in this section we are talking about those features that are not related directly to geo-spatial information yet we still considered important because we think that those features will somehow provide an implicit and indirect link of the causing factors of drivers'behavior, popularity and rank in the city and road network properties. We proposed 8 features in this category: the time of the day (in hours), whether it is weekday or weekend, the apartment rental price in the nearby area for those with 1, 2 and 3 bedrooms respectively, the temperature in the day and night, and the weather condition of the time (either sunny, cloudy, shower or rainy, etc.). It is quite straightforward for us to come up with these features, they can have linkage with the traffic situation. For example, the rental price of the nearby apartment reflects the popularity, population and land price of the area, which shall further imply the degree of importance and centrality in urban area. There are really some expensive residential areas in Shanghai like near the center of the city in Huangpu District, where the population is really high. The population density is a crucial matter that affects traffic. If the rental price is high, it means that there lives a high density of people, and as a result, the traffic vehicles that travels from and to this place will be massive, resulting in heavy traffic. So we think it is a significant feature that represents the population density of the area, the distribution over area data of which is hard for us to obtain.

Other features like the time of the day and the difference between weekdays and weekends are also self-explanatory: people have different goals and motivation of where they would like to drive to on weekdays and weekends, which will affect road network loads and as a result causing traffic problems. In rush hour of weekdays, the roads' connection where most people live and where the workplaces being located at are getting large flow of traffic. On weekends however, roads connecting the residential functional areas and shopping mall areas during the dinning time and the closing hour of the shopping mall will also become under heavy congestion. Other features like weather and temperature of the area are also straightforward and guaranteed to have positive effects on prediction results. When the weather is rainy and bad, by all our consensus, people tend to drive more than walking and taking public transit because of the bad weather makes people reluctant to spend time outside and rather to stay inside the cars. Also the speed of vehicle under bad weathers are lower than normal, contributing to the congestion. Extremely low and high temperatures will also cause the people drive more often because there is air conditioning inside and without the hassle and of walking outside.

3.6 Summary of Collected Features

After discussing all the three categories of features being extracted or referenced, we come to a summary of all the features we use in the next chapter, training and testing with the machine learning model. See Table 3.3 for reference on the feature list, with a total number of 311. Note that the weather condition feature is represented as a binary representation of 1-in-N representation, see Section 4.3.2 for details. Some of those features are input by user as prior knowledge during the prediction step. Though with different source and magnitude, either explicitly or implicitly representing the causing factors, they, being put together, comprise a large set of knowledge engineered by either we researchers as well as the computer algorithms, and that is one of the most important step and effort in application machine learning in real world scenarios.

Table 3.3 Feature list used in model training, testing and prediction

Number	Feature Name	Description
1	`price_1`	The average one-bedroom apartment rental price nearby.
2	`price_2`	The average two-bedroom apartment rental price nearby.
3	`price_3`	The average three-bedroom apartment rental price nearby.
4	`haveShop`	Boolean flag of whether there exists a shopping place nearby.

5 - 17	`roadType`	The type of the current road in tags from OSM database, choices mapped as 1-of-N representation. The road types include: `primary`, `pedestrian`, `secondary`, `path`, `demolished`, `construction`, `tertiary`, `service`, `track`, `trunk`, `residential`, `link`, `unclassified`.
18	`HouseHouseNum`	The number of optimal routes between residential areas themselves going through this segment.
19	`WorkWorkNum`	The number of optimal routes between business site areas themselves going through this segment.
20	`MallMallNum`	The number of optimal routes between shopping mall areas themselves going through this segment.
21	`HouseWorkNum`	The number of optimal routes from residential areas to business site areas going through this segment.
22	`HouseMallNum`	The number of optimal routes from residential areas to shopping mall areas going through this segment.
23	`WorkHouseNum`	The number of optimal routes from business site areas to residential areas going through this segment.
24	`WorkMallNum`	The number of optimal routes from business site areas to shopping mall areas going through this segment.

25	`MallHouseNum`	The number of optimal routes from shopping mall areas to residential areas going through this segment.
26	`MallWorkNum`	The number of optimal routes from shopping mall areas to business site areas going through this segment.
27	`time`	The time of the day in hours.
28	`temp`	Temperature at the current data instance's time slot.
29	`isweekend`	Boolean flag of whether the day is in weekend.
30 - 66	`currentNodeLeftX`	The number of POIs of each type around the crossing to the left of the current node.
67 - 103	`currentNodeRightX`	The number of POIs of each type around the crossing to the right of the current node.
104 - 140	`forwardOneCrossingX`	The number of POIs of each type around the first next crossing forward to the current node.
141 - 167	`forwardTwoCrossingX`	The number of POIs of each type around the second next crossing forward to the current node.
168 - 214	`forwardThreeCrossingX`	The number of POIs of each type around the third next crossing forward to the current node.
215 - 251	`backwardOneCrossingX`	The number of POIs of each type around the first next crossing backward to the current node.

252 - 288	`backwardTwoCrossingX`	The number of POIs of each type around the second next crossing backward to the current node.
289 - 325	`backwardThreeCrossingX`	The number of POIs of each type around the third next crossing backward to the current node.
324	`forwardCrossAvgDist`	The average distance between road crossings forward to the current node.
325	`backwardCrossAvgDist`	The average distance between road crossings backward to the current node.
326 - 342	`weather`	The weather condition in the area, choices mapped as 1-of-N representation. The condition string includes: `"clear"`, `"cloudy"`, `"flurries"`, `"fog"`, `"hazy"`, `"mostlycloudy"`, `"mostlysunny"`, `"partlycloudy"`, `"partlysunny"`, `"sleet"`, `"rain"`, `"snow"`, `"sunny"`, `"tstorms"`, `"unknown"`.

Chapter 4 Model and Test Result

4.1 Overview of Model Selection

The input of our model, if we abstract the real world problem into a mathematical model, is that we have all of our map and geographical data encoded inside the map, as well as other data sources for cross referencing and better describing the properties that matters. There is also a time variable and other temporal data that plays an important role. The output we desired from the model is the traffic situation, labeled in green, yellow, red and deep red, at the desired period of time at the location within given boundary.

From the description of our problem, it is clearly seen as a classification problem in machine learning. When the data are being used to predict a category, supervised learning is also called classification. This is the case when assigning a node or road segment with the predicted label of a 4 category traffic situation. When there are only two choices, this is called two-class or binomial classification. When there are more categories, this problem is known as multi-class classification, which is what we are tackling now, more precisely a 4-class classification problem.

There are quite a number of classification algorithms available, including Naïve Bayes, Logistic Regression, Decision Trees and Support Vector Machines (SVMs). There are also lots of combinations of those algorithms into ensemble method that also outperforms a single classifier. We try to select the best suitable model for our traffic prediction problem. Considerations when choosing an algorithm mainly includes:

- **Accuracy.** Getting the most accurate answer possible isn't always necessary. Sometimes an approximation is adequate, depending on what you want to use it for. If that's the case, you may be able to cut your processing time dramatically by sticking with more approximate methods. Another advantage of more approximate methods is that they naturally tend to avoid overfitting.

- **Training time.** The number of minutes or hours necessary to train a model varies a great deal between algorithms. Training time is often closely tied to accuracy —one typically accompanies the other. In addition, some algorithms are more sensitive to the number of data points than others. When time is limited it can drive the choice of algorithm, especially when the data set is large.

- **Linearity.** Lots of machine learning algorithms make use of linearity. Linear classification algorithms assume that classes can be separated by a straight line (or its higher-dimensional analog). These include logistic regression and support vector machines. Linear regression algorithms assume that data trends follow a straight line. These assumptions aren't bad for some problems, but on others they bring accuracy down. Despite their dangers, linear algorithms are very popular as a first line of attack. They tend to be algorithmically simple and fast to train.

- **Number of parameters.** Parameters are the knobs a user gets to turn when setting up an algorithm. They are numbers that affect the algorithm's behavior, such as error tolerance or number of iterations, or options between variants of how the algorithm behaves. The training time and accuracy of the algorithm can sometimes be quite sensitive to getting just the right settings. Typically, algorithms with large numbers parameters require the most trial and error to find a good combination. Alternatively, there is a parameter sweeping way that automatically tries all parameter combinations at whatever granularity you choose. While this is a great way to make sure you've spanned the parameter space, the time required to train a model increases exponentially with the number of parameters. The upside is that having many parameters typically indicates that an algorithm has greater flexibility. It can often achieve very good accuracy. Provided you can find the right combination of parameter settings.

- **Number of features.** For certain types of data, the number of features can be very large compared to the number of data points. This is often the case with genetics or textual data. The large number of features can bog down some learning algorithms, making training time unfeasibly long. Support Vector Machines are particularly well suited to this case, and in our problem there are more than 300 features.

In the process that we deal with finding out the best model suitable for our data and the problem, which is mainly test, cross-validation or hold-out process, we used a way to find out which model and which set of features works the best. In summary the whole process is typically a cycle: design a set of features, run an experiment and analyze the results on a validation data set, calculate the score, change the feature set or the machine learning model and start over.

Finally, after doing experiments with the famous ensemble method AdaBoost and linear classifier SVM. AdaBoost Freund and Schapire (1997), a popular boosting algorithm, core principle of which is to fit a sequence of weak learners (i.e., models that are only slightly better than random guessing, such as small decision trees) on repeatedly modified versions of the data. The predictions from all of them are then combined through a weighted majority vote (or sum) to produce the final prediction. The data modifications at each so-called boosting iteration consist of applying weights $w_1, w_2, ..., w_N$ to each of the training samples. Initially, those weights are all set to $w_i = 1/N$, so that the first step simply trains a weak learner on the original data. For each successive iteration, the sample weights are individually modified and the learning algorithm is reapplied to the reweighted data. At a given step, those training examples that were incorrectly predicted by the boosted model induced at the previous step have their weights increased, whereas the weights are decreased for those that were predicted correctly. As iterations proceed, examples that are difficult to predict receive ever-increasing influence. Each subsequent weak learner is thereby forced to concentrate on the examples that are missed by the previous ones in the sequence Hastie et al. (2005).

The results from the comparison experiment show that AdaBoost does not outperform SVM in this problem. So we choose to use the family of SVM classifiers because of its fast speed and relatively good accuracy in the context of linear as well as its extensive resources and popularity.

4.2 Support Vector Machines (SVMs)

Support vector machines (SVMs) Cortes and Vapnik (1995) are a set of supervised learning methods used for classification, regression and outliers detection. The advantages of support vector machines are:

- Effective in high dimensional spaces.
- Still effective in cases where number of dimensions is greater than the number of samples.
- Uses a subset of training points in the decision function (called support vectors), so it is also memory efficient.
- Versatile: different Kernel functions can be specified for the decision function. Common kernels are provided, but it is also possible to specify custom kernels.

A support vector machine constructs a hyper-plane or set of hyper-planes in a high or infinite dimensional space, which can be used for classification, regression or other tasks. Intuitively, a good separation is achieved by the hyper-plane that has the largest distance to the nearest training data points of any class (so-called functional margin), since in general the larger the margin the lower the generalization error of the classifier. Given training vectors $x_i \in \mathbb{R}^p, i = 1, ..., n$, in two classes, and a vector $y \in \{1, -1\}^n$, SVC solves the following primal problem:

$$\min_{w,b,\zeta} \frac{1}{2} w^T w + C \sum_{i=1}^{n} \zeta_i \tag{4-1}$$

$$\text{subject to } y_i(w^T \phi(x_i) + b) \geq 1 - \zeta_i, \tag{4-2}$$

$$\zeta_i \geq 0, i = 1, ..., n \tag{4-3}$$

Its dual is

$$\min_{\alpha} \frac{1}{2}\alpha^T Q\alpha - e^T\alpha \tag{4-4}$$

$$\text{subject to } y^T\alpha = 0 \tag{4-5}$$

$$0 \le \alpha_i \le C, i = 1, ..., n \tag{4-6}$$

where e is the vector of all ones, $C > 0$ is the upper bound, Q is an n by n positive semidefinite matrix, $Q_{ij} \equiv y_i y_j K(x_i, x_j)$ where $K(x_i, x_j) = \phi(x_i)^T \phi(x_j)$ is the kernel. Here training vectors are implicitly mapped into a higher (maybe infinite) dimensional space by the function ϕ.

The decision function is:

$$\text{sgn}(\sum_{i=1}^{n} y_i \alpha_i K(x_i, x) + \rho) \tag{4-7}$$

SVC, NuSVC and LinearSVC are variants of SVMs capable of performing multi-class classification on a dataset. SVC and NuSVC implement the "one-against-one"approach Knerr et al. (1990) for multi-class classification. If n_class is the number of classes, then $n\_class * (n\_class - 1)/2$ classifiers are constructed and each one trains data from two classes. On the other hand, Linear SVC implements "one-vs-the-rest"multi-class strategy, thus training n_class models. If there are only two classes, only one model is trained. In our problem, we choose to use linear SVC as a multi-class classifier. The kernel function can be any of the following:

- linear: $\langle x, x' \rangle$
- polynomial: $(\gamma \langle x, x' \rangle + r)^d$
- rbf: $\exp(-\gamma |x - x'|^2)$
- sigmoid $(\tanh(\gamma \langle x, x' \rangle + r))$

4.3 Training Models

4.3.1 LIBSVM and LIBLINEAR

Until now, we already have all the data collected and processed into features using the feature extraction program that we introduced in the previous chapter. We followed certain schema of data and types and finally we should prepare all the extracted features into a data format that could be required and recognized by LIBSVM and LIBLINEAR programs. They are open source software and very robust in real world performance. These implementations of support vector machine algorithm is widely used and tested, so we choose to use them in our own project.

LIBSVM Chang and Lin (2011) is an integrated software for support vector classification, (C-SVC, nu-SVC), regression (epsilon-SVR, nu-SVR) and distribution estimation (one-class SVM). It supports multi-class classification. Since version 2.8, it implements an SMO-type algorithm Fan et al. (2005). LIBSVM provides a simple interface where users can easily link it with their own programs. Main features of LIBSVM include: different SVM formulations, efficient multi-class classification, cross validation for model selection, probability estimates, various kernels (including precomputed kernel matrix), weighted SVM for unbalanced data, etc.

LIBLINEAR Fan et al. (2008) is a linear classifier for data with millions of instances and features. It supports the following classifiers:

- L2-regularized classifiers
- L2-regularized classifiers
- L2-loss linear SVM, L1-loss linear SVM, and logistic regression (LR)
- L1-regularized classifiers
- L2-loss linear SVM and logistic regression (LR)
- L2-regularized support vector regression
- L2-loss linear SVR and L1-loss linear SVR.

A good advantage of LIBLINEAR is that it uses the same data format as LIBSVM, the general-purpose SVM solver introduced above, and also similar usage. Multi-class classification is done by either 1) one-vs-the rest or 2) Crammer & Singer. In our problem, we use the one-vs-the-rest for multi-class classification.

The difference between the two libraries are quite obvious too. The main idea is that LIBLINEAR is optimized to deal with linear classification (i.e. no kernels necessary), whereas linear classification is only one of the many capabilities of LIBSVM, so logically it may not match up to LIBLINEAR in terms of classification accuracy. Also, when there are some large data for which with/without nonlinear mappings gives similar performances. Without using kernels, one can quickly train a much larger set via a linear classifier. In such suitable cases, the cross-validation time is significantly reduced by using LIBLINEAR. Our traffic prediction problem also uses the LIBLINEAR as a model training tool and a prediction tool because we have large amount of data and the high performance, especially the real-time data prediction ability is crucial in our online application.

4.3.2 Processing Extracted Features

Categorical Feature

SVM requires that each data instance is represented as a vector of real numbers. Hence, if there are categorical attributes, we first have to convert them into numeric data. We recommend using m numbers to represent an m-category attribute. Only one of the m numbers is one, and others are zero. For example, in our case, we have a feature representing weather conditions. Since all the weather conditions are described as words and no real-numbered value is meaningful, or can be compared with each other in terms of values while representing such categorical feature. So it is a typical problem that could be seen as a 15-category attribute, with all the possible weather conditions in our weather data. It is therefore represented as (0,0,...,0,1), (0,0,...,1,0), (0,...,1,0,0) and so on. This is the so-called one-in-N representation of features, with each field having a binary value. Our experience indicates that if the number of values in an attribute is not too large, this coding might be more stable than using a single number.

Scaling

Scaling before applying SVM is very important. Part 2 of Sarle *et al.* (1997) explains the importance of this and most of the considerations also apply to SVM. The main advantage of scaling is to avoid attributes in greater numeric ranges dominating those in smaller numeric ranges. Another advantage is to avoid numerical difficulties during the calculation. Because kernel values usually depend on the inner products of feature vectors, e.g. the linear kernel and the polynomial kernel, large attribute values might cause numerical problems. We recommend linearly scaling each attribute to the range $[-1, +1]$ or $[0, 1]$. Of course we have to use the same method to scale both training and testing data. For example, suppose that we scaled the first attribute of training data from $[-10, +10]$ to $[-1, +1]$. If the first attribute of testing data lies in the range $[-11, +8]$, we must scale the testing data to $[-1.1, +0.8]$. The LIBSVM provides a helpful utility to help scale and normalize the input feature data ranges, with the scaling parameter outputs saved to a file which should be later read by the prediction program. This could be done by the following command, calculate and applying the same scaling parameters in both generating scaled data in range [0, 1] for training and testing and it is important for accuracy:

```
$ ../svm-scale -l 0 -s scaling_parameter train > train.scale
$ ../svm-scale -r scaling_parameter test > test.scale
```

4.3.3 Imbalanced Dataset

In our problem, one of the biggest caveats that emerges is the class imbalance issue. Imbalanced data typically refers to a problem with classification problems where the classes are not represented equally. In our case, there would be always more data representing the class of clear traffic (green) than other classes, often to a large ratio, because traffic jam happens only from time to time. As a result, the model trained will prefer to predict every instance of test data into the major representing class, because that would achieve a high accuracy, which is not right if the model simply predict all the traffic situation into “good”, which is a big part of the occurrences. They would cause many problems, like we cannot use accuracy anymore to perform the optimal parameter search of the model with accuracy as goal, because the imbalanced distribution would greatly shift the result to one-class sided. The accuracy paradox is the case where your accuracy measures tell the story that you have excellent accuracy (such as 90% in our case), but the accuracy is only reflecting the underlying class distribution. It is very common, because classification accuracy is often the first measure we use when evaluating models on our classification problems. To compare solutions, we will use alternative metrics (True Positive, True Negative, False Positive, False Negative) instead of general accuracy of counting number of mistakes. In order to solve this, there are several ways to achieve the goal. We can roughly classify the approaches into two major categories: sampling based approaches and cost function based approaches:

Sampling based approaches

This can be roughly classified into three categories:

- Oversampling, by adding more of the minority class so it has more effect on the machine learning algorithm
- Undersampling, by removing some of the majority class so it has less effect on the machine learning algorithm
- Hybrid, a mix of oversampling and undersampling

However, these approaches have clear drawbacks. By undersampling, we could risk removing some of the majority class instances which is more representative, thus discarding useful information. By oversampling, just duplicating the minority classes could lead the classifier to overfitting to a few examples.

Table 4.1 Imbalanced distribution of training set, caused by the intrinsic anomaly nature of traffic situations, resolved by assigning a weight to each class based on number of representations

Class	#Occurrence	Percentage	Weight
Green (Good)	2197252	0.90953	1
Yellow (Slow)	176517	0.07306	14
Red (Congested)	40629	0.01681	59
Deep Red (Extremely Congested)	1105	0.00046	2186

Cost function based approaches

The intuition behind cost function based approaches is that if we think one false negative is worse than one false positive, we will count that one false negative as, e.g., 100 false negatives instead. For example, if 1 false negative is as costly as 100 false positives, then the machine learning algorithm will try to make fewer false negatives compared to false positives (since it is cheaper). In case of SVM, different classes can have different weights on them, resulting in the desired loss penalty Osuna et al. (1997). For imbalanced data sets we typically change the mis-classification penalty per class. This is called class-weighted SVM, which minimizes the following:

$$\min_{\mathbf{w},b,\xi} \quad \sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j \kappa(\mathbf{x}_i,\mathbf{x}_j) + C_{pos}\sum_{i\in\mathcal{P}}\xi_i + C_{neg}\sum_{i\in\mathcal{N}}\xi_i, \tag{4-8}$$

$$s.t. \quad y_i\Big(\sum_{j=1}^{N}\alpha_j y_j \kappa(\mathbf{x}_i,\mathbf{x}_j) + b\Big) \geq 1-\xi_i, \qquad i=1\ldots N \tag{4-9}$$

$$\xi_i \geq 0, \qquad i=1\ldots N \tag{4-10}$$

where $\mathcal{P}$ and $\mathcal{N}$ represent the positive/negative training instances. In standard SVM we only have a single $\mathcal{P}$ value, whereas now we have 2. The misclassification penalty for the minority class is chosen to be larger than that of the majority class. Essentially this is equivalent to oversampling the minority class: for instance if $C_{pos} = 2C_{neg}$ this is entirely equivalent to training a standard SVM with $C = C_{neg}$ after including every positive twice in the training set. See Table 4.1 for the weight put on each class based on occurrences of small classes.

SMOTE

In 2002, a sampling based algorithm called SMOTE (Synthetic Minority Over-Sampling Technique) Chawla et al. (2002) was introduced that try to address the class imbalance problem. It is one of the most adopted approaches due to its simplicity and effectiveness. It is a combination of oversampling and undersampling, but the oversampling approach is not by replicating minority class but constructing new minority class data instance via an algorithm.The intuition behind the construction algorithm is that oversampling causes overfit because of repeated instances causes the decision boundary to tighten. Instead, we will create "similar"examples instead. To the machine learning algorithm, these new constructed instances are not exact copies and thus softens the decision boundary as a result. As a result, the classifier is more general and does not overfit.

4.3.4 Training Process

Our prepared training set consists of data collected from 19 May to 25 May, a time period of 7 days with weekdays and weekends. So as a test purpose training data, it is well covered the situations and our extracted features, even though the data number is still relatively small, not being able to make the model robust enough. Also the data set is too imbalanced due to the very nature of the problem, we could only begin to solve this after collecting much more data so that the minor classes could have their amount and representatives without being overwhelmed. Each line of the data file for LIBLINEAR is a line representing a data instance. A data instance of a classification problem is often represented in the following form.

```
label feature 1, feature 2, . . ., feature n
```

After scaling is done, we split and sampled the dataset into training set and test set for later validation. The split is done by selecting the data lines that are previous in the time as training data, and use those with a more recent time as test data. By construction data set this way, we could test out how good the training data from the historical time points would perform in predicting traffic situation at a future time. Alternatively, we could split the data so that the data in one district of Shanghai are used as training set and those in another district are used as test set. By splitting data this way, we could evaluate how good it will perform to predict traffic situation with new geospatial information using the old ones from another place. Or we could just randomly sample the dataset to create a training set and a test set. Considering the data imbalance issues, that is, the labels' distribution is not balanced, because there would always be more roads in green state than in deep red state, causing the score computations and accuracy invalid. So we have to sample a balanced number of data instances that is equivalent to the data lines with the least number of label occurrence.

After training data and testing data are created, it is the time to feed the organized and scaled training data into the `train` program to train a model. There are a few parameters that we could tune for the LIBLINEAR, the most important of which is the type of solver to use, switched by the parameter `-s`. The following are all the available solvers for multi-class classification, which includes both Logistic Regression and SVM:

```
0 -- L2-regularized logistic regression (primal)
1 -- L2-regularized L2-loss support vector classification (dual)
2 -- L2-regularized L2-loss support vector classification (primal)
3 -- L2-regularized L1-loss support vector classification (dual)
4 -- support vector classification by Crammer and Singer
5 -- L1-regularized L2-loss support vector classification
6 -- L1-regularized logistic regression
7 -- L2-regularized logistic regression (dual)
```

Formulations for the above available classification solvers:

For L2-regularized logistic regression (-s 0), we solve

$$min_w w^T w/2 + C \sum log(1 + exp(-y_i w^T x_i)) \tag{4-11}$$

For L2-regularized L2-loss SVC dual (-s 1), we solve

$$min_{\alpha} 0.5(\alpha^T(Q + I/2/C)\alpha) - e^T\alpha \tag{4-12}$$

$$s.t. \quad 0 <= \alpha_i, \tag{4-13}$$

For L2-regularized L2-loss SVC (-s 2), we solve

$$min_w w^T w/2 + C\sum max(0, 1 - y_i w^T x_i)^2 \tag{4-14}$$

For L2-regularized L1-loss SVC dual (-s 3), we solve

$$min_{\alpha} 0.5(\alpha^T Q\alpha) - e^T\alpha \tag{4-15}$$

$$s.t. \quad 0 <= \alpha_i <= C, \tag{4-16}$$

For L1-regularized L2-loss SVC (-s 5), we solve

$$min_w \sum |w_j| + C\sum max(0, 1 - y_i w^T x_i)^2 \tag{4-17}$$

For L1-regularized logistic regression (-s 6), we solve

$$min_w \sum |w_j| + C\sum log(1 + \exp(-y_i w^T x_i)) \tag{4-18}$$

For L2-regularized logistic regression (-s 7), we solve

$$min_{\alpha} 0.5(\alpha^T Q\alpha) + \sum \alpha_i * log(\alpha_i) + \sum(C - \alpha_i) * log(C - \alpha_i) - constant \tag{4-19}$$

$$s.t. \quad 0 <= \alpha_i <= C, \tag{4-20}$$

$$\text{where } Q \text{ is a matrix with } Q_ij = y_i y_j x_i^T x_j. \tag{4-21}$$

The LIBLINEAR implements 1-vs-the rest multi-class strategy for classification, which our problem used. In training i vs. non_i, their C parameters are weight from $-wi * C$

and C, respectively.

After several experiments with different solvers, we selected the L2-regularized L2-loss support vector classification (primal) as the solver. Primal-based solvers (Newton-type methods) are moderately fast in most situations compared to the dual-based solvers (coordinate descent methods). In contrast, it is sometimes suggested to use the dual-based solvers when dealing with large sparse data (e.g., documents) under suitable scaling and C is not too large, or data with number of instances much less than the number of features. In our traffic prediction problem, it is clear that it does not belong to these two situations.

4.4 Testing Result and Evaluation

After training the training set using LIBLINEAR, we obtained the model which could be used to test the scores on the test set, by comparing the predicted labels with the correct labels. We are able to easily predict a test set by using the `predict` program, also provided by LIBLINEAR. Note that the test data set are also being scaled with the same scaling parameter as the training set, that is, the test data is on the same data scale and range with the testing data on each feature. The test data set is collected from 28 May to 1 June, so that compared to the training set, the test set is the future time data, which could correctly and scientifically measure and test our model's performance. In our case, the values' ranges between features vary largely due to the different type and aspect of the feature extraction and data source, and such scaling will help with the accuracy.

As stated in the previous section, our problem is mainly dealing with telling rare classes that happens from one very large class. Classification and evaluation such imbalanced classes are real challenges, because accuracy no longer representing the model's correctness effectively, thus the optimization of parameter in common algorithms are not reliable anymore using accuracy as metric.

In order to assess the model trained, we would introduce three scoring and evaluation methods use: Precision, Recall and F1 metrics.

Precision-Recall metric are usually used to evaluate classifier output quality. In information retrieval, precision is a measure of result relevancy, while recall is a measure of how many truly relevant results are returned. A high area under the curve represents both high recall and high precision, where high precision relates to a low false positive rate, and high recall relates to a low false negative rate. High scores for both show that the classifier is returning accurate results (high precision), and returning a majority of all positive results (high recall).

A system with high recall but low precision returns many results, but most of its predicted labels are incorrect when compared to the training labels. A system with high precision but low recall is just the opposite, returning very few results, but most of its predicted labels are correct when compared to the training labels. An ideal system with high precision and high recall will return many results, with all results labeled correctly.

Precision (P) is defined as the number of true positives (T_p) over the number of true positives plus the number of false positives (F_p).

$$P = \frac{T_p}{T_p + F_p} \tag{4-22}$$

Recall (R) is defined as the number of true positives (T_p) over the number of true positives plus the number of false negatives (F_n).

$$R = \frac{T_p}{T_p + F_n} \tag{4-23}$$

These quantities are also related to the (F_1) score, which is defined as the harmonic mean of precision and recall.

$$F1 = 2\frac{P \times R}{P + R} \tag{4-24}$$

It is important to note that the precision may not decrease with recall. The definition of precision ($\frac{T_p}{T_p+F_p}$) shows that lowering the threshold of a classifier may increase the denominator, by increasing the number of results returned. If the threshold was previously set too high, the new results may all be true positives, which will increase precision. If the previous threshold was about right or too low, further lowering the threshold

Table 4.2 Precision, recall and F1 scores of the test result

Class	Precision	Recall	F1 Score	Support
Green (Good)	0.93	0.93	0.93	716106
Yellow (Slow)	0.24	0.10	0.14	53039
Red (Congested)	0.07	0.14	0.09	9400
Deep Red (Extremely Congested)	0.00	0.24	0.01	215
Avg / Total	0.87	0.87	0.87	778760

will introduce false positives, decreasing precision. Recall is defined as $\frac{T_p}{T_p+F_n}$, where $T_p + F_n$ does not depend on the classifier threshold. This means that lowering the classifier threshold may increase recall, by increasing the number of true positive results. It is also possible that lowering the threshold may leave recall unchanged, while the precision fluctuates.

See Table 4.2 for the scores of three different performance metrics the test result with popular way features representing the non-local geospatial features introduced in Section 3.4, and also with correct and identical data set scaling. From the scores of result data, we can see how big the impact of data imbalance issue can be. As seen in the table, the overall scores for predicting traffic that is good, which is the most popular class, is really out performing those for other minor classes, which is as high as 0.93. The scoring metrics are taking the distribution into consideration, and we can see that our system predict poorly for heavy traffic situations, especially the red and deep red ones. It is seen here that while doing cross-validation can help optimize our model, the imbalance issue would also break the evaluation. For imbalanced data sets, accuracy may not be a good criterion for evaluating a model. It may be better to conduct cross-validation and prediction with respect to different criteria (F-score, AUC, etc.). We achieved this model by feature engineering worked by trying and testing with our without the set of feature, and compare various metrics, not limited to just accuracy, to make feature selection and achieve near optimal model. The tables also show that normalization and regularization of data set is crucial for linear SVC, which is sensitive to varying ranges. Besides, getting the scaling parameter right and consistent are crucial in such an application with large varying feature value ranges.

We also collected and sampled some data points at certain time points from Baidu

Map's traffic prediction functionality. We sampled the prediction at a historical time point, and check whether the prediction is correct or not at the same location when the future time come. We wish to set up a baseline for our application to compare against. Though the problem abstraction between ours and Baidu Map's are a little different, since we mainly consider the affects of the geospatial surroundings on traffic, while Baidu Map uses more likely temporal historical data at the location to make prediction, it can still shed some light on us for our future progressions. We then tested our model by predicting the test data collected several days after when the training set is collected, and in the mean time, we compared the prediction result and accuracy at the same district, Huangpu, Shanghai. We reached an accuracy of 86.747% while Baidu Map has an accuracy of 82.798%. In Table 4.3, we could see the prediction result evaluation of our model compared to the traffic prediction data collected from Baidu Map on day and hour periods basis. The count of data instances are aggregated into time periods. Readers may note that the total number of data instances of Baidu Map traffic prediction is always larger than ours, it is because that though both of them covered the whole Huangpu District, the points sampled for our test data is a bit smaller than the points sampled by Baidu Map, where the points were selected in a rectangular bounding box, which consists of some marginal data outside Huangpu District.

Besides from the relatively close and good prediction rate during non-rush hours or weekdays, we perform sometimes better at rush hours where traffic anomalies appears more often. It is seen that most of the time our test result using mere linear SVC model exceeds the Baidu Map's traffic prediction result and accuracy based on time-series historical data, which is a great evidence that feature engineering on both local and non-local geospatial data indeed contribute to the model. It is also seen that at rush hours on workdays, the prediction accuracy is significantly lowered due to the model not performing so well while classifying extremely small classes, which is heavy traffic jam. Also, one should also notice that there are some test time periods that our system perform much worse than the Baidu Map traffic prediction, like the accuracy at 17 pm, May 31 is as low as 65.327%, while the accuracy of Baidu Map at this time is 70.620%, and like the accuracy at 20-23 pm, May 31 is also showing a large accuracy drop-down as Baidu Map's is 82.841% and ours is 70.925%. It is apparently anomalies in our data set, since the trend of accuracy simply changed a lot at this time, compared with the accuracy at the other days at the same time. We still need to investigate more on this issue, and looking deep into the prediction data, we found that our model performs really bad while having lots of traffic jams happening in the area which is generally low in probability. We still need to collect more and more data, since currently we only used 7 days of traffic data, which is far from enough for the model to be performing well against such anomalies when sudden outliers or events like holiday, extreme weather or traffic accident happens. That means our current model is sensitive and not robust enough.

Table 4.3 The prediction result evaluation of our model compared to the traffic prediction data collected from Baidu Map, and all testing and comparison were done within the area of Huangpu District, Shanghai, and test set divided and predicted on hourly basis.

Date	Hour	Baidu Map Traffic Prediction			Our Test Result		
		#Correct	#All	Accuracy(%)	#Correct	#All	Accuracy (%)
5.28	16-17	38465	52393	73.416	18366	24330	75.487
	18	24730	34920	70.819	12688	16211	78.268
	19-23	54783	69869	78.408	24336	32449	74.998
5.29	0-8	193174	208941	92.454	93692	97396	96.197
	9-11	72256	87335	82.734	35445	40540	87.432
	12-17	61488	87323	70.414	34914	40545	86.112
	18-19	38158	52407	72.811	21820	24344	89.632
	21-23	73472	87338	84.124	37061	40562	91.369
5.30	0-7	143761	156830	91.667	69386	73051	94.983
	10-14	82890	104798	79.095	39897	48659	81.993
	15-16	38132	52398	72.774	19986	24316	82.193
	19	11831	17462	67.753	6958	8111	85.785
	20-23	89219	104812	85.123	41987	48677	86.256
5.31	0-7	95963	104387	91.930	46137	48711	94.716
	9	26245	34940	75.114	13422	16217	82.765
	10-14	80915	104797	77.211	37976	48655	78.052
	17	12333	17464	70.620	5300	8113	65.327
	18-19	23927	34928	68.504	11710	16222	72.186
	20-23	43418	52411	82.841	17259	24334	70.925
6.1	0-7	126991	139389	91.105	59625	64923	91.840
	9	14564	17469	83.371	6919	8090	85.525
	10-13	40160	52402	76.638	20665	24304	85.027
Overall		1386875	1675013	82.798	675549	778760	86.747

Chapter 5 Implementation

5.1 System Architecture

As a working system that is capable of handling user requests and make real time predictions, an appropriate system architecture is required. Thanks to the open source project OpenStreetMap, we can borrow most of its architecture since it is real world tested and robust and rigid. First of all, the database we used are the same as the OpenStreetMap server map database, which is PostgreSQL. The reason for selecting this database software for all of our crucial map data is because it is really one of the fastest and most state-of-art database, most importantly with strong support of geospatial database extension, PostGIS. Such extension comes extremely handy with some functions in SQL to help ease the pain of querying spatial data as well as spatial index for speedups. The backend for providing map editing (with frontend application in JavaScript called iD), showing slippy map, and map API are from the OpenStreetMap's Rails Port project writing in Ruby on Rails language framework (`https://github.com/openstreetmap/openstreetmap-website`). We simply run an instance of this backend at our own server, and it uses our own database for data source. The OSM API is a REST web service interface for reading and writing to the database i.e. XML over HTTP, with use of simple URLs for object access, and standard HTTP response codes. Other OSM components access the database via this interface. It is also available to the outside internet. The API logic is all part of the same Ruby on Rails application which powers the OSM front end website. Another part of our system are the map rendering engine that serves on our server the latest map in form of tile map images, so that the fronted could have beautiful cartography. The main backend for the rendering of the maps that are produced from OSM data in the PostGIS instance of the database is open source software Mapnik (`http://mapnik.org/`). There are several great open source utilities provided by OpenStreetMap that is really useful for converting and processing the downloaded data from data dump sites and used in scheduled jobs to exporting and importing up-

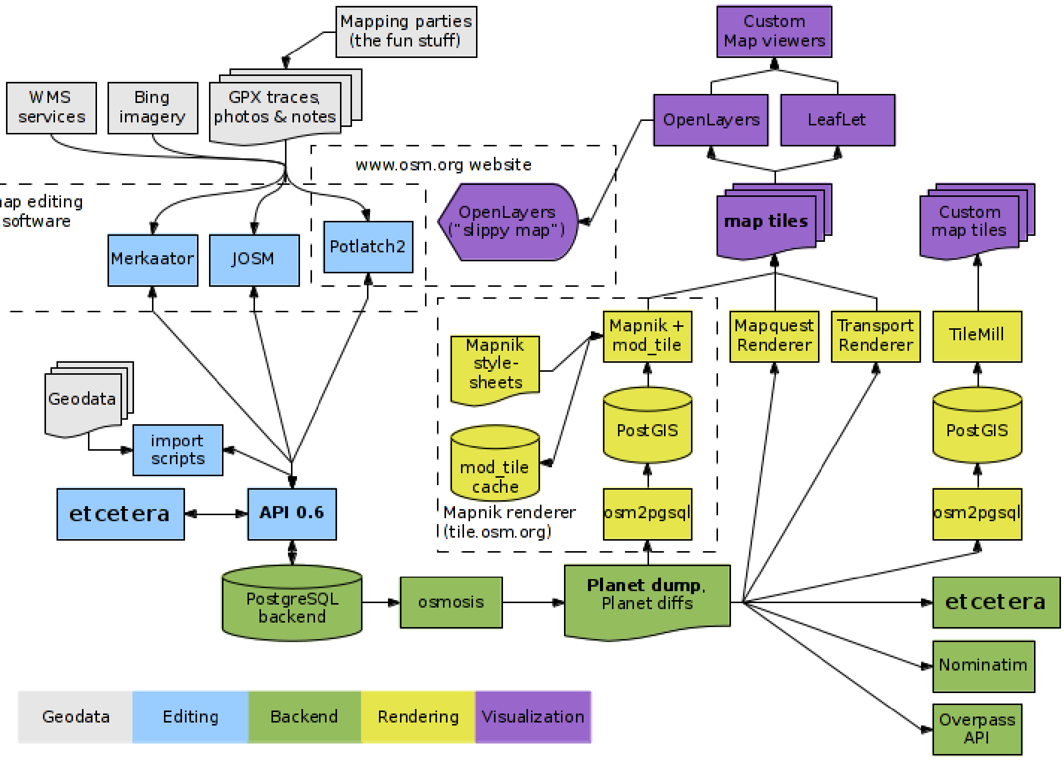

Figure 5.1 OpenStreetMaps software stack and components

dated data from API database to the PostGIS database used by Mapnik for tile rendering, which include Osmosis, an OSM data processing Swiss army knife, and OSM data importer for rendering or geo-coding, osm2pgsql, a powertool for importing OSM XML files into PostGIS databases. Finally, there is our own written prediction engine which provides the prediction API for real-time requests for predicting traffic behaviors with provided parameters and previously trained machine learning model. It is written in Java as a servlet application running inside Tomcat container. A demo web frontend page is also online, using the popular Leaflet.js library (`http://leafletjs.com/`) which is our main demo showcase. The server where we are running all these system components is powered by Ubuntu 12.04.5 LTS, with Intel Xeon(R) CPU E5504 CPU and 48 GB of memory.

5.2 Map Database

The map database is one of the most important part in our system because it stores all the information we previously obtained. The map database is also called the main database since obviously it is where we keep all our data. The main database is accessed for editing via the API, so the editing is also made in this database. The database contains tables for each element type (nodes, ways, relations). In fact, for each of these there are several database tables: current, history, current_tags, history_tags. In addition, there are database tables for storing changeset, gpx_files, users, diary entries, sessions, oauth etc. The database contains several tables and relations that is shown it the figure. The database we used is PostgreSQL, with strong geospatial capabilities. PostgreSQL has geometry types. For our core OSM database we do not use these. We have own representation of OpenStreetMap data primitives. The PostGIS extension for PostgreSQL is often used for geographic data. PostGIS adds geospatial functions and two metadata tables. Again we do not use this for our core database, however we do use all of these things on the tile server database as required by the Mapnik rendering engine. Osmosis can be used to populate a more general PostgreSQL/PostGIS database from a OSM data dump file, where we initially populated and loaded the Baidu POIs. We also added some other geographic related functions and data types to our instance of database, such as proximity querying, etc. A changeset, as you can see in both database schemas and OSM XML files, consists of a group of changes made by a single user over a short period of time, for example someone edited the map in a session of urban planning process. One changeset may for example include the additions of new elements to OSM, the addition of new tags to existing elements, changes to tag values of elements, deletion of tags and also deletion of elements. The ER-diagram is shown in Figure 5.2, 5.3 and 5.4, the most important tables are *current_nodes/nodes, current_ways/ways, current_node_tags/node_tags* as well as *current_way_nodes/way_nodes*. Those are the table we used most frequently because they hold our map data primarily. Those columns named k and v are just representing the tags in the form of key-value pairs. Several of indexes and foreign key on columns on tables exist, refer to the DDL statement for more

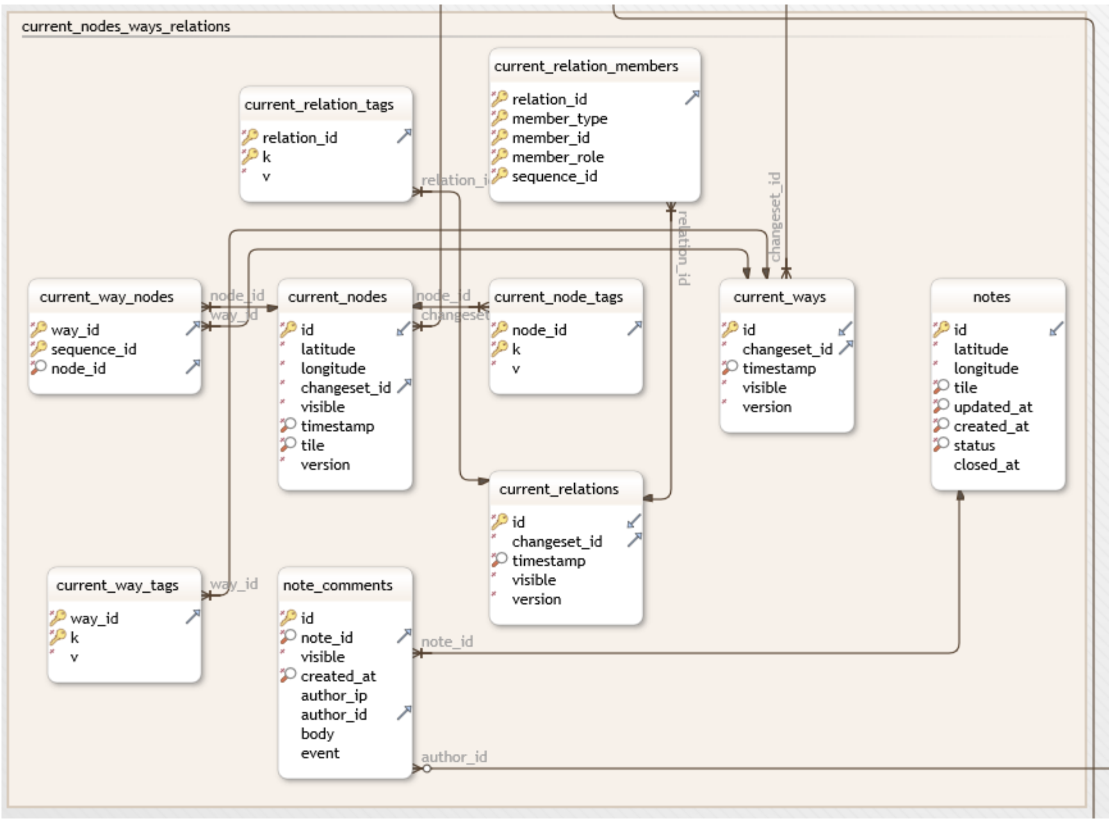

Figure 5.2 Main database ER-diagram part 1

details.

5.3 Tile Rendering

The process of rendering a map generally means taking raw geospatial data and making a visual map from it. Often the word applies more specifically to the production of a raster image, or a set of raster tiles, but it can refer to the production of map outputs in vector-based formats. "3D rendering" is also possible taking map data as an input. The ability to render maps in new and interesting styles, or highlighting features of special interest, is one of the most exciting aspects having open access to geo-data. Developers in and around the OpenStreetMap community have created a wide variety of software for rendering OpenStreetMap data. The data can also be converted to other data formats for use with existing rendering software. The rendering engine we used to provide those beautiful tile maps is called Mapnik. Mapnik is an open source toolkit for rendering

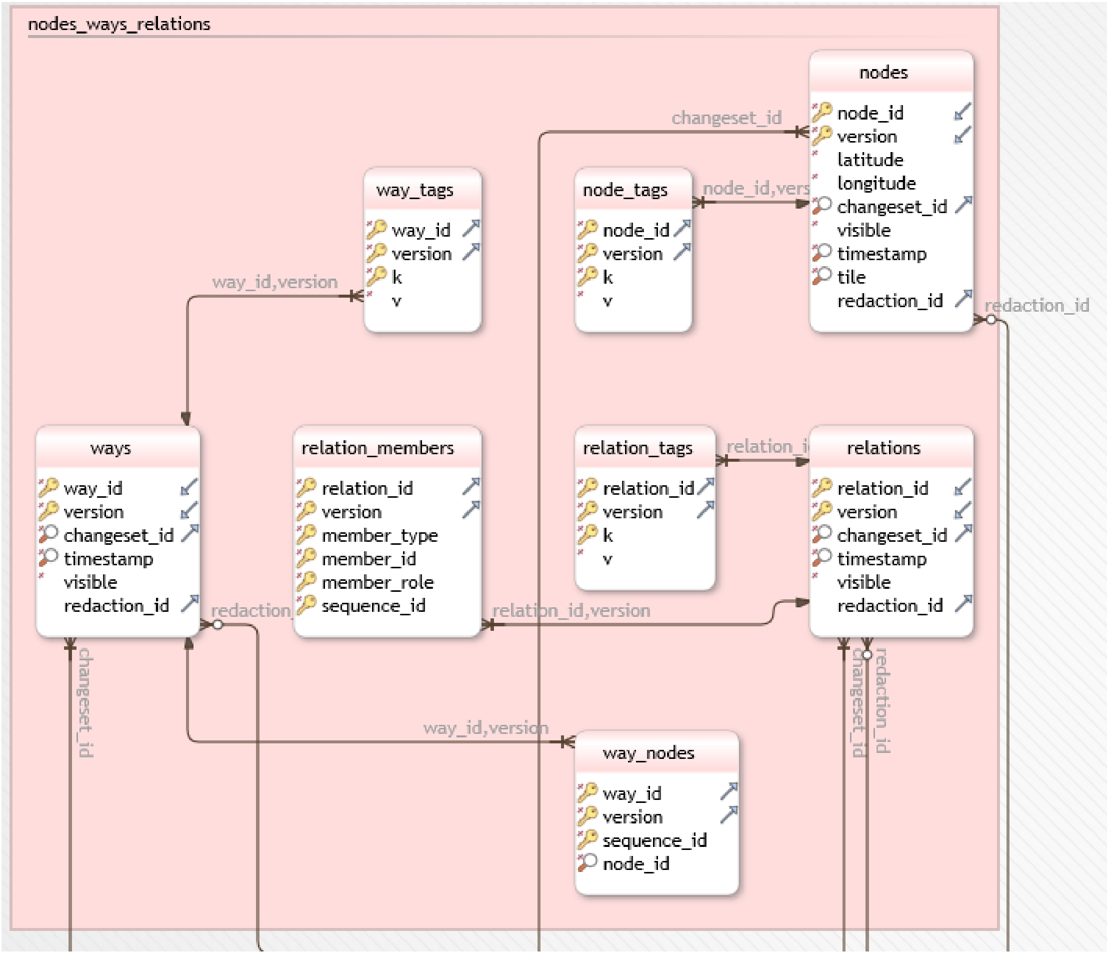

Figure 5.3 Main database ER-diagram part 2

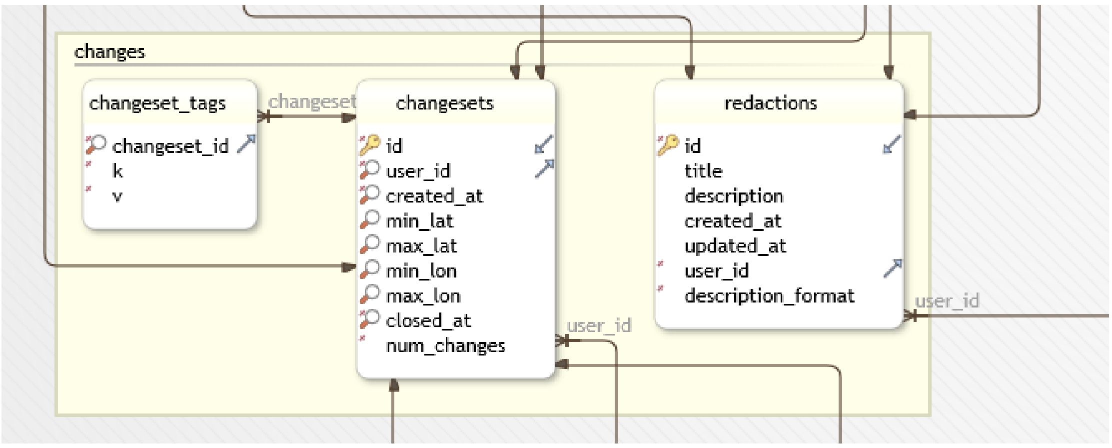

Figure 5.4 Main database ER-diagram part 3

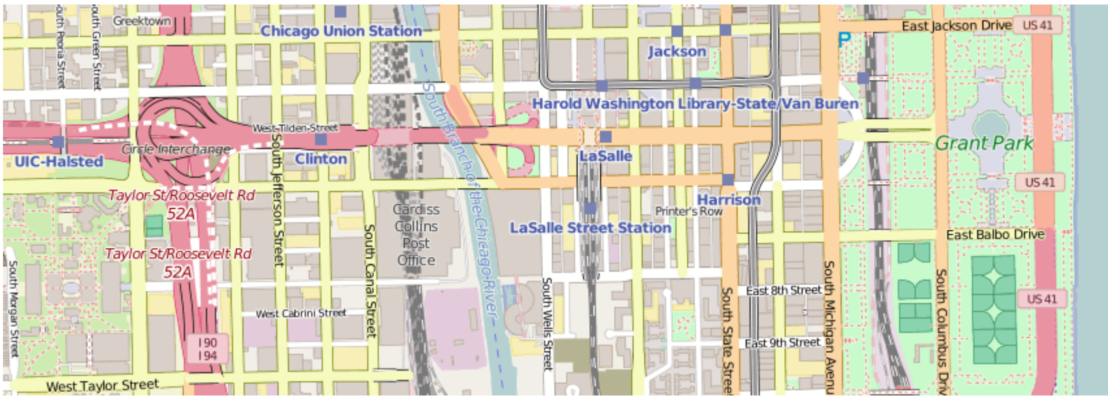

Figure 5.5 An example rendering using Mapnik with OpenStreetMap carto style

maps. Among other things, it is used to render the five main Slippy Map layers on the OpenStreetMap website. It supports a variety of geospatial data formats and provides flexible styling options for designing many different kinds of maps. Mapnik is written in C++ and can be scripted using binding languages such as JavaScript (Node.js), Python, Ruby, and Java. It uses the AGG rendering library and offers anti-aliasing rendering with subpixel accuracy. Mapnik can use data from different sources: it can directly process OSM data, PostGIS databases, shapefiles and more. In our system design, PostGIS database is used, with data updated from the main database we introduced earlier. PostGIS is the most common approach for rendering OSM data with Mapnik. OSM can be loaded by a tool such as osm2pgsql or Imposm and accessed via SQL queries and GIS functions defined in a Mapnik style. This approach can be used for more advanced renderings, and is the main datasource used by the Standard OpenStreetMap layer. Mapnik allows for customization of all the cartographic aspect of a map - data features, icons, fonts, colors, patterns, and even certain effects such as pseudo-3d buildings and drop shadows. This is all controlled by defining datasources and style rules, most commonly in an XML language specific to Mapnik. The style we used is the same as the OpenStreetMap standard one, which is open source and beautiful (written in CartoCSS supported by Mapnik, `https://github.com/gravitystorm/openstreetmap-carto`). The Figure 5.5 shows the style used.

5.4 Prediction Engine

This is the most important part in the whole project and is very crucial and special. It is a servlet program written in Java, as a Servlet and running in the very popular Tomcat Servlet container. The prediction engine listens at a HTTP port and accept requests with given parameter. It is running with URL pattern /prediction and with query parameters x1, x2, y1, y2 representing the bounding box longitude and latitude of the prediction area, as well as the hour representing the hour of the day ranging from 0 to 24, and price representing the average house rental price of the requested area, along with day_temp and night_temp, representing the temperature of day and night. After some calculations and processing, the prediction engine responds with a large JSON format data, containing list of 5-tuples, each with the form of (starting coordinates, ending coordinates, predicted traffic situation). Every two consecutive nodes in one way will form such a tuple representing a road segment, the smallest unit of prediction. In cases of one-way road, only one tuple exists for one such segment, and in cases of two-way road, two tuples exist for one such segment, with starting coordinates and ending coordinates exchanged, and with new predicted traffic situation.

During this process, the program mainly does a few things in steps to be clear. Firstly, the Servlet program called prediction.java receive the request and parse the parameter described above, getting the bounding box and some other designated parameters for prediction. Then the program access the main map database introduced above, querying the *current_nodes, current_ways* and *current_way_nodes* table with the condition that only nodes and ways within the bounding box get selected. After successfully getting the query result, the map data is stored in an object with all the connections and relations as Map data structure in Java. A method named organizeWays is called to organize queried ways and nodes into the right sequence and become connected. Then begins the data preprocessing to achieve the input data in required format for LIBSVM. First generate the crossings of the roads and calculate all the features related to crossings by the class named genCross and calCross. We calculate the crossings several blocks ahead and behind, following the same rule that was used in previous training and testing experiments, storing data in feature result list. After that, the program begins to search and count the number of points of interests of all types predefined and derived from Baidu types. As stated in data collection chapter, all the POIs have been imported into the main database's *current_nodes/nodes* table with specific tag poitype=*, so with the help of the strong and robust spatial functionality that PostgreSQL database provides, we can use SQL queries to easily search all the points within a given radius of the given coordinates. It is achieved by Cube and EarthDistance, and these 2 extensions provide easy to use and very fast methods to accomplish some more minor geo related activities. Before doing anything, we prepared the database with two lines of SQL to create such extensions:

```
CREATE EXTENSION cube;
CREATE EXTENSION earthdistance;
```

Listing 5.1 Create required extension in PostgreSQL

Query and get a set of all POI types that exist in our database:

```
SELECT * FROM current_node_tags WHERE current_node_tags.k='poitype';
```

Listing 5.2 Query all POI types

And then, at the beginning of each request, we create a temporary table called poi_nodes out of only POI nodes that have necessary tags inside, so that the search later would be in a much smaller range of data.

```
CREATE TEMPORARY TABLE poi_nodes ON COMMIT PRESERVE ROWS AS SELECT *
    FROM current_nodes WHERE current_nodes.id IN (SELECT
    current_node_tags.node_id FROM current_node_tags;
```

Listing 5.3 Create temporary table poi_nodes

And we created a spatial index using GiST on latitudes and longitudes of each node for it on the fly. GiST stands for Generalized Search Tree. It is a balanced, tree-structured access method, that acts as a base template in which to implement arbitrary indexing schemes. B-trees, R-trees and many other indexing schemes can be implemented in GiST. The advantage of GiST is that it allows the development of custom data types with the appropriate access methods, by an expert in the domain of the data type, rather than a database expert. So we can be sure that what we are querying is all about longitude and latitude. Though making this index in context of the whole Shanghai area would take more than one or two seconds, it is worth the trade off because after my experiment, with index each radius search query would take nearly 2 seconds to complete the sequential search and heavy calculating each row. After the indexing, each query takes about 20ms to complete, which is a great improvement. Note that the longitude and latitude in the database are being multiplied by 1e7 as an integer, so here we shall divide it back to normal double.

```
CREATE INDEX on poi_nodes USING gist(ll_to_earth(poi_nodes.latitude *
    1.0 / 1e7, poi_nodes.longitude * 1.0 / 1e7));
```

Listing 5.4 Create index on table poi_nodes

Then comes the interesting part. Because we have a lot of nodes from all ways in the area, the search is quite extensive. Each node that belongs to a way has a feature with the counting of how many points of interest of given type are there in vicinity of the node, and add those numbers to the feature list of every node. This is where the previously created extension comes useful. The two of the many functions we use here is calculating the distance between coordinates and finding records in a radius. To calculate the distance between 2 coordinates we use *earthdistance(lltoearth($latlngcube), lltoearth($latlng_cube))*. This function allows us to pass 2 sets of coordinates and it will return a numerical value representing meters. Another great function provided by these extensions is *earth_box(ll_to_earth($lat, $lng), $radius_in_metres)*. This function allows us to perform a simple compare to find all records in a certain radius. This is done by the function by returning the great circle distance between the points. The following is the actual query that run for each node and each type to be counted. The searching radius is 200 meters in our case.

```
SELECT * FROM poi_nodes WHERE earth_box(ll_to_earth(lat, lon), radius
   ) @> ll_to_earth(poi_nodes.latitude * 1.0 / 1e7, poi_nodes.
   longitude * 1.0 / 1e7);
```

Listing 5.5 Query the POIs in range

After iterating through all the nodes to query the points of interest around each given node. We managed to get those POI related features calculated and stored in format for later use.

The next step in the whole prediction process is to prepare those calculated and queried features into prediction test data in order. It iterates through all the ways that previously visited in order, and look up relevant nodes within each way, extracting all the related items and features. For the first iteration process, it deals with the forward direction and in the second it deals with the backward direction. During the process, each feature line is written into a text file in the original order preserved so that later on the prediction process could output readable results. It is basically a file with many lines as the whole test data set, each line contains node ID and all the features used. Here is the end of the feature extraction process.

When the next step of the real prediction starts, firstly the data set scale procedure provided from the LIBSVM library is invoked, with the same scaling parameters as the ones used in the training data set and model so that the consistency of data normalization could be achieved. After scaling all the testing data's feature value right, the predict program from the open source LIBLINEAR is called for high speed prediction of the test data with given SVM model file. It takes the normalized data input for best result and the output is redirected to a text file. This process is actually relatively fast due to the high performance. It cost less time than the most time consuming data processing and feature extraction steps.

As for the last step of returning and responding the predicted traffic situation of all roads back to the client, the prediction Servlet needs to read the test result and read the result in the same order as before when it is written. Because of this, it is trivial to save those predicted results along with the nodes' information and the consecutive nodes' segments for later drawing in the frontend. Now we simply attach the starting coordinates and the ending coordinates together with the predicted traffic on this segment, and the one-way and two-way situations are treated separately. The case of the two-way roads actually becomes really tricky because we need the frontend to draw distinguishable lines of possibly different color (the traffic situation on each direction of the roads may vary largely due to many issues). The lines could not just have their starting coordinate and ending coordinate exchanged, because that would result in an exactly overlapped two lines in the same place with different color, causing the user not being able to distinguish. To solve this, we have to make a little bit of the offset to the second instance of the road segment representing the opposite direction. That is, when coming across the backward direction, we calculate the starting and ending coordinates using the linear parallel offset formula as we consider the longitude/latitude coordinates is equivalent to Descartes Coordinates in such a small area on earth. We neglected the fact that the earth' s surface is a sphere. By offsetting 1 to 2 meters distance to the original segment vector, the result is already distinguishable between two directions.

However, regardless of those variants, the final result is a JSON file containing a list object containing a list of all the 5-tuples to be drawn. The response is then sent back the requesting client for later process. This is all the process that our prediction engine would do, it is still quite slow when adding all this together and one of our future work is to improve the response time.

5.5 Frontend Showcase

Each application needs a good and useful interface to function properly. In our frontend page, it is made up of only one HTML file, along with several open source JavaScript libraries. The appearance is not fancy nor extensive for users, but it shows some of our system's most import usage, which is predicting under the condition of current data and time and weather. The editing interface has a separate page and link, currently not customized and we simply use the iD online OSM map editor provided by OpenStreetMap and included in the Rails Port project which we have been running on our own server and introduced earlier. The demo page of the prediction we wrote is based on a very popular open source online mapping library called Leaflet.js (`http://leafletjs.com/`) which could be able to handle our own customized tile server and draw the prediction results in each color as polylines on the tile map layer. Leaflet is the leading open-source JavaScript library for mobile-friendly interactive maps. Weighing just about 33 KB of JS, it has all the mapping features most developers ever need. Leaflet is designed

Figure 5.6 Frontend demo page

with simplicity, performance and usability in mind. It works efficiently across all major desktop and mobile platforms, can be extended with lots of plugins, has a beautiful, easy to use and well-documented API and a simple, readable source code that is a joy to contribute to, and that is why we choose to use it because it is really strong and small in size. On the page there are several input boxes and dropdown selections that allows the user to input the scalar parameters that needs to be provided by the user. Below the input form is where the Leaflet library comes into use. There is a slippy map interface that allows the user to freely zoom in and zoom out as well as pan around to browse the map. The tile image layer is defined while creating the class with our own Mapnik rendering server. After the user put the desired area into the screen size, he or she would click on the predict button with all the parameter input. The frontend code would retrieve the longitude and latitude of the bounding box that the map currently shows, checking the zoom to see if it is too big an area that would cause potential crash. Then, along with all the input data after checking validity, all those parameters are retrieved from the frontend page and the page would send a AJAX HTTP request to the prediction engine's listening URL. This will initiate a prediction process and while waiting for response,

Figure 5.7 Frontend demo prediction result page

the frontend web page shows up a notification popup modal indicating that it is currently running. After the data is received and all the road segments, three new multi-polyline layers are initialized and pushed all lines with different prediction results represented in different colors defined earlier into the layer's lists respectively. The lines contained the longitude and latitude starting and end point data, inserted into the layer with the right color by switching on the predicted result. Then the layers loaded with data are attached to the map object, so that the result could be shown on the web page in correct colors. A quick note is that all the traffic prediction results is drawn on the client side in browser by JavaScript. It turns vector data format into the images, so the performance requirement is actually quite heavy, so limiting the rendering area is necessary. This explains the whole process that the frontend does, basically only calling the backend prediction engine to get the test result and draw the result on top of the map by using the Leaflet.js library's tile layers and vector layers'functionality. See Figure 5.6 for user interface and Figure 5.7 for a small area prediction test.

Chapter 6 Conclusion

To conclude, this project contains not only the data mining and machine learning algorithms used to extract the features, train the model and predict the traffic situation with given input features, but also includes the server interface and web application that one can edit, browse and calculate the prediction result as an extra layer on map.

For the algorithm and model part, we show at last that our feature and model as well as the system is capable of predicting future traffic at a relatively good accuracy with road network, points of interest and other spatial data, along with temporal and other related non-spatial data as input. We also made a comparison against Baidu Map's traffic prediction accuracy, at the same area and same time period, with the data points sampled, collected and divided into hourly basis and into time slots. We exceeded Baidu Map most of the time in terms of accuracy. Besides from the relatively close prediction during non-rush hours or weekdays, we perform much better at rush hours where traffic anomalies appears more often. We use class-weighted SVM for training to compensate for the extreme data imbalance issues. Our special point and the advantage over other similar traffic prediction researches or production is that we can accept a variable map as an input, that is we take more consideration into the properties that lies in the geographical spatial data maps, rather than predicting by merely using historical traffic data and time-series models which is prevalent in this field. We tested by changing and selecting features and confirmed on the positive effect that how both local geographical features like POI or road density and non-local features like population distribution, traveling behavior and routing preferences, etc. affects the traffic flow in urban areas.

As for the platform we developed, the demo is already fully functional at the time, available at `http://adapt.seiee.sjtu.edu.cn:8080/traffic/` and users can browse the map in Shanghai, making edits or doing prediction with input parameters and current bounding box, then the prediction result would show up as a map overlay with colors representing traffic on the road. Our system which consists of a lot of services that communicates and process with each other is fully set up, and we designed the system to be robust and decoupled enough so that if we later manage to improve or model, the interchangeable model file can be replaced and the system will instantly show a better result. Meanwhile, our crawlers to get history data as well as testing and training data from various sources are still up and running, continuing the data collection and training.

We still have some plans of future work, though. Our prediction scores is really bad for minor classes like the anomalies like congested traffic situations, due to the intrinsic nature of extreme imbalance. So first of all is continuing working on the data set and class representation imbalance issue to greatly increase our performance while dealing with heavy traffic hours. There are promising methods like synthetic data resampling, SMOTE, as described in earlier Section 4.3.3 as well as using techniques used in anomaly detection such as one-class SVM and decision trees, since some traffic situations are very much like anomalies. We shall investigate more on the data collection quality as well as feature engineering to train a better model with higher accuracy. We will find more data sources relevant to the traffic issues that may become useful because cross-referencing is important. Time-series methods like Markov process or moving average methods are also worth a try. Dig deeper into the human mobility and migration behaviors, especially in urban environments, by implementing simulation process in an abstracted network or graph with traffic being described as flow, is also a promising way to improve. Currently our demo's prediction speed is not fast enough for conveying a good user experience. Besides from the interface updates, we plan to optimize the prediction speed by implementing offline feature extraction which is the most costly part, as well as speed up the algorithm by using distributed computing platform and database. Online machine learning algorithms and models are preferred because while we continuously crawl the data, we would like to make the model better and take feedback into account from newly-become historical data.

REFERENCE

[1] Çolak S, Lima A, González M C. Understanding congested travel in urban areas.[J]. Nature communications, 2016, 7:10793. http://www.nature.com/ncomms/2016/160315/ncomms10793/full/ncomms10793.html.

[2] Simini F, González M C, Maritan A, et al. A universal model for mobility and migration patterns[J]. Nature, 2012, 484(7392):96–100. https://arxiv.org/ftp/arxiv/papers/1111/1111.0586.pdf.

[3] Schnitzler F, Liebig T, Mannor S, et al. Combining a gauss-markov model and gaussian process for traffic prediction in dublin city center[J]. CEUR Workshop Proceedings, 2014, 1133(2):373–374.

[4] Qi Y, Ishak S. A Hidden Markov Model for short term prediction of traffic conditions on freeways[J]. Transportation Research Part C: Emerging Technologies, 2014, 43:95–111. http://dx.doi.org/10.1016/j.trc.2014.02.007.

[5] Ren Y, Ercsey-Ravasz M, Wang P, et al. Predicting commuter flows in spatial networks using a radiation model based on temporal ranges.[J]. Nature communications, 2014, 5:5347. http://www.ncbi.nlm.nih.gov/pubmed/25373437.

[6] Xu J, Deng D, Demiryurek U, et al. Context-aware online spatiotemporal traffic prediction[J]. IEEE International Conference on Data Mining Workshops, ICDMW, 2015, 2015-Janua(January):43–46.

[7] Xu J, Deng D, Demiryurek U, et al. Mining the Situation: Spatiotemporal Traffic Prediction with Big Data[J]. IEEE Journal on Selected Topics in Signal Processing, 2015, 9(4):702–715.

[8] Pan B, Demiryurek U, Shahabi C. Utilizing real-world transportation data for accurate traffic prediction[J]. Proceedings - IEEE International Conference on Data Mining, ICDM, 2012:595–604.

[9] Mchugh D. Traffic Prediction and Analysis using a Big Data and Visualisation Approach[J]. 2015.

[10] Xu Y, Kong Q J, Klette R, et al. Accurate and interpretable bayesian MARS for traffic flow prediction[J]. IEEE Transactions on Intelligent Transportation Systems, 2014, 15(6):2457–2469.

[11] Horvitz E J, Apacible J, Sarin R, et al. Prediction, Expectation, and Surprise: Methods, Designs, and Study of a Deployed Traffic Forecasting Service[J]. Conference on Uncertainty in Artificial Intelligence, 2012:1–10. http://arxiv.org/abs/1207.1352.

[12] Coast S. OpenStreetMap[M].[S.l.]: [s.n.] , 2004. http://www.openstreetmap.com.

[13] Frey B J, Dueck D. Clustering by Passing Messages Between Data Points[J]. Science, 2007, 315:972–976. www.psi.toronto.edu/affinitypropagation.

[14] Pedregosa F, Varoquaux G, Gramfort A, et al. Scikit-learn: Machine learning in Python[J]. The Journal of Machine Learning Research, 2011, 12:2825–2830.

[15] Dijkstra E W. A note on two problems in connexion with graphs[J]. Numerische mathematik, 1959, 1(1):269–271.

[16] Hart P E, Nilsson N J, Raphael B. A Formal Basis for the Heuristic Determination of Minimum Cost Paths[J]. IEEE Transactions on Systems Science and Cybernetics, 1968, 4(2):100–107.

[17] Freund Y, Schapire R E. A decision-theoretic generalization of on-line learning and an application to boosting[J]. Journal of computer and system sciences, 1997, 55(1):119–139.

[18] Hastie T, Tibshirani R, Friedman J, et al. The elements of statistical learning: data mining, inference and prediction[J]. The Mathematical Intelligencer, 2005, 27(2):83–85.

[19] Cortes C, Vapnik V. Support-vector networks[J]. Machine learning, 1995, 20(3):273–297.

[20] Knerr S, Personnaz L, Dreyfus G. Single-layer learning revisited: a stepwise procedure for building and training a neural network[M]//Neurocomputing.[S.l.]: Springer, 1990:41–50.

[21] Chang C C, Lin C J. LIBSVM: a library for support vector machines[J]. ACM Transactions on Intelligent Systems and Technology (TIST), 2011, 2(3):27.

[22] Fan R E, Chen P H, Lin C J. Working set selection using second order information for training support vector machines[J]. The Journal of Machine Learning Research, 2005, 6:1889–1918.

[23] Fan R E, Chang K W, Hsieh C J, et al. LIBLINEAR: A library for large linear classification[J]. The Journal of Machine Learning Research, 2008, 9:1871–1874.

[24] Sarle W S, et al. Neural network FAQ[J]. Periodic posting to the Usenet newsgroup comp. ai. neural-nets, 1997.

[25] Osuna E, Freund R, Girosi F. Support vector machines: Training and applications[J]. 1997.

[26] Chawla N V, Bowyer K W, Hall L O, et al. SMOTE: synthetic minority oversampling technique[J]. Journal of artificial intelligence research, 2002:321–357.

Acknowledgments

I would like to express my deepest gratitude to my advisor Prof. Kenny Zhu, who has always been there to listen and give practical advice. His patience, enthusiasm, great kindness and immense knowledge helped me in all the time of research. Whenever I encountered difficulties and become frustrated, Prof. Zhu would be always willing to contribute his valuable feedback, advice and encouragement, illuminating the way for me to continue to search for the answers. I could not have asked for a better advisor. In addition to the academic collaboration, I greatly value Prof. Zhu's guidance on the personal integrity and the rigorous attitude toward research. Without his consistent instructions, this thesis could not be in this shape.

I am also thankful to ADAPT Lab for providing wonderful academic facilities and atmosphere. In particular, many thanks to my friends Qi Lu, Jianan Wang, Yifei Huang in ADAPT Lab for numerous discussions on related topics and collaborations. It's my pleasure to work with you during these years.

My gratitude would go to my beloved family as well as my girlfriend, Tingting He, for their consistent encouragement and loving considerations during these years, wherein I gained great confidence in developing a good research work.