一、上次的讨论：

- 从论文中找原因，弄清楚实验数据中，出发的网格id格式、目标网格id、网格变化是如何处理的来的
- 对预计时间进行随机高斯采样

二、原文中提及实验的部分：

一、本文主要目标：

> Assuming that there are three candidate grids Di (0-2), the main problem is to choose a most appropriate grid for drivers to hunt passengers. Based on the above, we take the hot degree of Di, the road condition of O to Di, and the average earning of Di into consideration comprehensively, and produce a final score for every grid to express the selection degree. **Finally the grid Di scoring the most will be chosen and recommended to taxi drivers.**

二、对地图的网格划分：

> In the module of offline mining and training, we first handle the map (map meshing) and the GPS trajectory big data (trajectories segmenting), respectively. Unlike road networks used in most of existing approaches, in this paper, we use the grid map generated from map meshing. After trajectories segmenting, we carry out the task of map matching. That is, each GPS point is attached to the corresponding road. Thereby, we can easily achieve the whole occupied or idle trips.(p4)

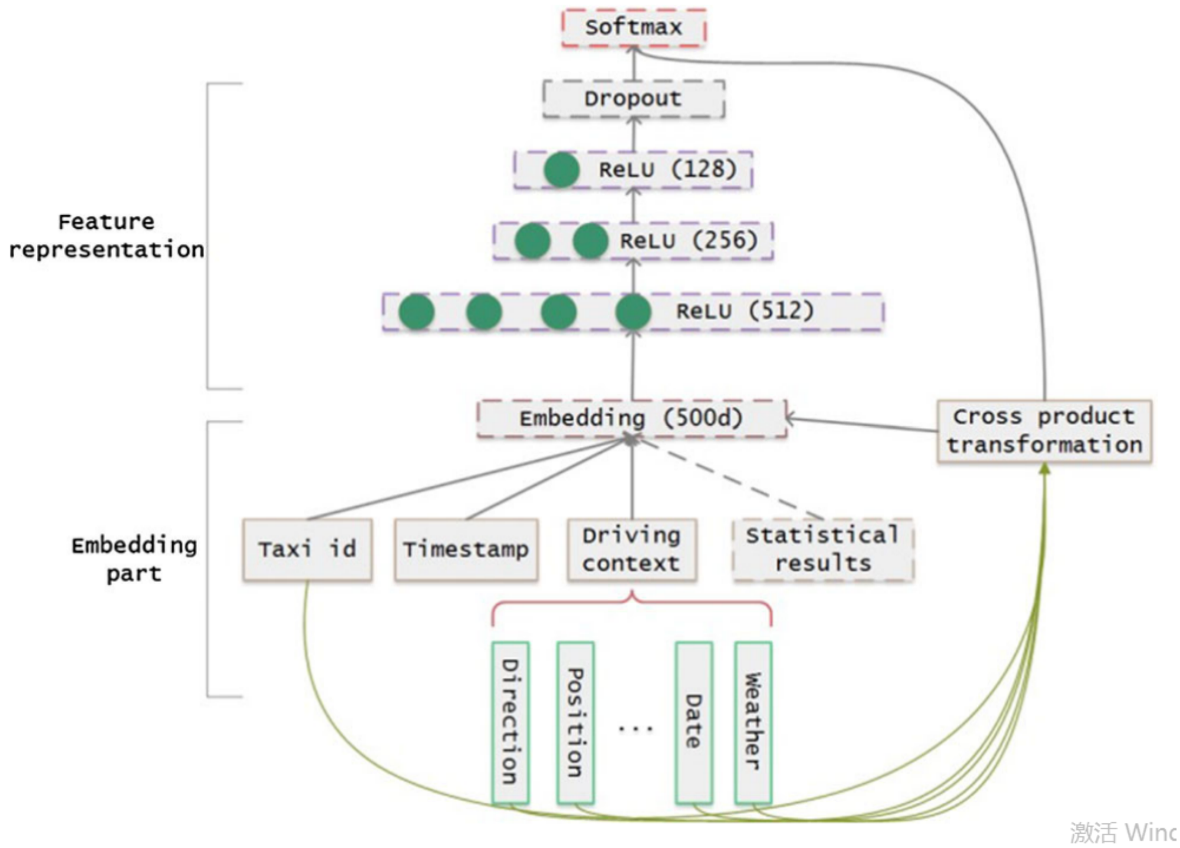
三、对数据的规范化预处理：

> After this step, we utilize statistical learning approaches to handle the grip map and trips obtained in the front and produce three types of complete information: (1) picking up points, (2) road information, and (3) the earning of each grid. These three types of information are the **main input data of the wide & deep models.** It is worth noting that the trips obtained in the front cannot be straightly used in the wide & deep models. And we need to preprocess these trips and extract the discriminative features from them. Then these data pass through **normalization (for continuous features)** and **encoding operations (for categorical features)** to guarantee effectiveness and reasonableness. After this step, we employ the word embedding methods, such as word2vec or GloVe, to transform these features to lowdimensional vectors. Followed by word embedding, two efficient algorithms (sparse autoencoder and t-SNE) are used for dimensionality reduction.

四、wide & deep model

key information：

- crossed features are applied as higher-order features by Cartesian product.



激活 Winc

五、 Evaluation for TRec

how they evaluate their model?

> TRec combines these three components with weights and calculate a score for every grid in the map. Then TRec recommends the grid which has the highest score to taxis. What the taxi drivers most concern about is their earnings eventually.
>
> we use the following formula to calculate a score for every grid g in the map（p11）：
>
> $$\Upsilon(g) = \frac{\sqrt{w_{pp} \times w_{ge}}}{w_{rc}} \times \frac{\Psi \times \Gamma}{D} \times \lambda^{\lfloor \frac{k}{10} \rfloor}.$$

example:

> In order to evaluate our recommendation system TRec, we choose 1568 trips which are all produced between 5 am and 11 pm. These trips are chosen following the normal distribution. According to the charge standard of taxis in Shanghai, we utilize the following equation to achieve the earnings of a specific trip T, where |T| is the length of T.

三、GitHub代码对数据的描述：

出租车id	出发网格坐标	出发方向	出发时间	预计寻客时间	网格变化	目标网格id
14276	69.0-14.0	d	9.6	1050s	-7.0x2.0y	1677
80862	15.0-29.0	a	17.4	6s	0.0x0.0y	2378

1. ~~出发方向：用字母a-i分别表示从当前网格到下一网格的8个方向~~
2. ~~出发时间：用时刻(24进制)+分钟/60~~
3. ~~预计寻客时间：以秒为单位~~
4. **网格变化：当前网格到目标网格的变化情况**
5. ~~目标网格id：选择一个起点坐标对全局地图进行划分，按照经纬度整理为同等大小的网格~~

四、代码对数据的处理

代码对数据的处理的默认输入，就是带有以上特征的数据

util下的代码解析：

- new_devide.py:大数据集分割为小数据集
- myTime.py：时间戳处理
- geo_dis:计算两点距离
- **g_train_data.py**：计算**出发网格坐标**，**网格变化**
- direction.py：计算八个方向
- dig.py：处理切割后的数据，对每一出租车计算其寻客情况数据，数据写入pre.txt

```
def data_format(fname):
    i = 0
    fr = open(fname, 'r')
    fw = open('input' + repr(size) + '.txt', 'a')
    for line in fr:
        # string
        line=line.rstrip('\n')
        tid,s_lon,s_lat,s_time,e_lon,e_lat,e_time = line.split(',')
        s_x, s_y = grid_index(base, float(s_lon), float(s_lat), size)
        e_x, e_y = grid_index(base, float(e_lon), float(e_lat), size)
        angle = direction_index(s_x, s_y, e_x, e_y)
        des=e_x*101+e_y-1
        angle=chr(angle+97)
        e_x-=s_x
        e_y-=e_y

        print (i)
        print (e_time)
        i=i+1
        if len(s_time)<=2:
            s_time="000"+s_time
        if len(e_time) <= 2:
            e_time="000"+e_time

        s_t = time_transfer(s_time)
        e_time=e_time.rstrip()
        last_t=time_duration(s_time,e_time)
        #筛除潜在异常数据点
        if last_t<'60':
            continue
        #tmp = ','.join([repr(s_x), repr(s_y), repr(s_t), repr(e_x),
repr(e_y), repr(e_t)])
        tmp = ','.join([tid, repr(s_x)+'-'+repr(s_y), angle, repr(s_t),
last_t,repr(e_x)+'x'+repr(e_y)+'y',repr(des)])
        fw.write(tmp + '\n')
```

```
if __name__ == '__main__':
    fname = sys.argv[1]
    data_format(fname)
```