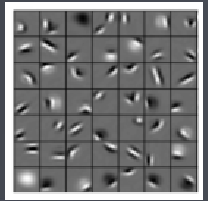
Place Category Prediction using Graph Convolutional Model

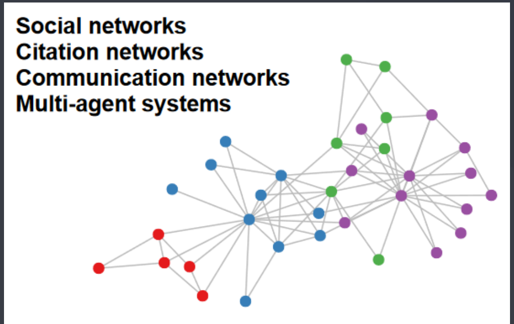
Soojung Hong

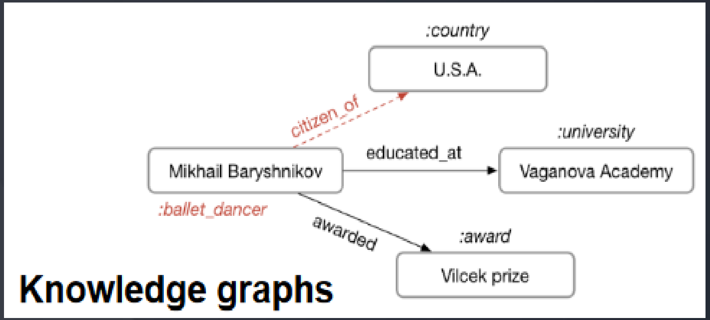
September 27. 2019

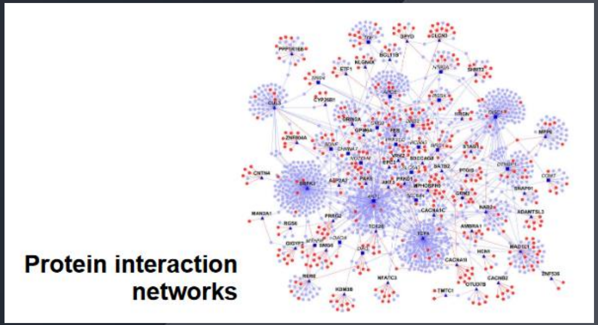
Why Graph Model in Deep Learning?



- A lot of real-world data does not "live" on grids (Thomas Kipf)
- Learn representations that encode structural information
- Standard DNN, CNN model didn't handle structure
- Mapping the geometric relationship in the learned space





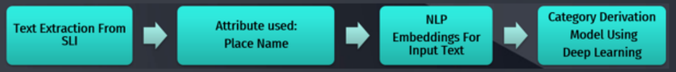


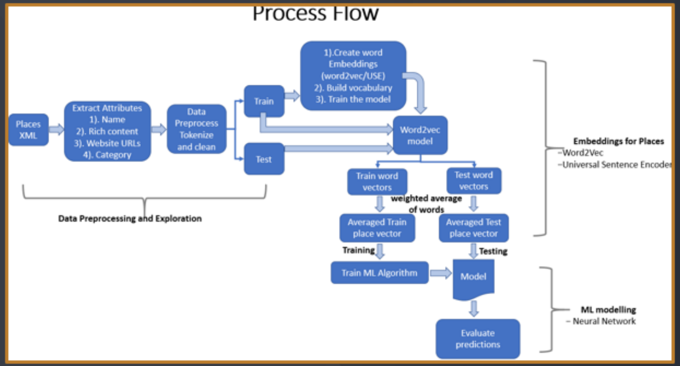
Graph structured data
(http://tkipf.github.io/misc/SlidesCambridge.pdf)

Why Graph Model in HERE Category Prediction problem?

- Lack of features about places

Category Derivation Model





	precision	recall	f1-score
100	0.75	0.74	0.74
200	0.61	0.57	0.59
300	0.85	0.80	0.83
350	0.89	0.90	0.89
400	0.75	0.70	0.72
500	0.76	0.65	0.70
550	0.71	0.71	0.71
600	0.61	0.58	0.60
700	0.55	0.51	0.53
800	0.63	0.70	0.66
900	0.58	0.75	0.66
avg / total	0.69	0.68	0.69

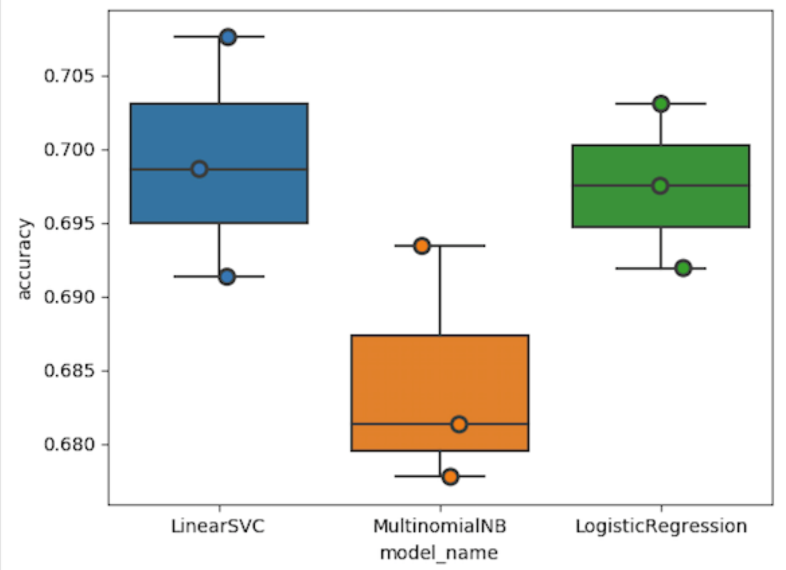
Level 1 : ~ **68%** accuracy
Level 1 – Level 2 : **60%**
Level 1 – Level 2 – Level 3 : ??

- Training data : 88,000 samples per category
- Test data : 15,000 samples per category
- Neural network with 3 hidden layers, adam optimizer, softmax output layer

Credit : Charitha H.

Limit : Category Prediction using only Place name

Q : Can we predict place category based on place (POI) name?
A : Yes, With approximately 70% accuracy. But,,,



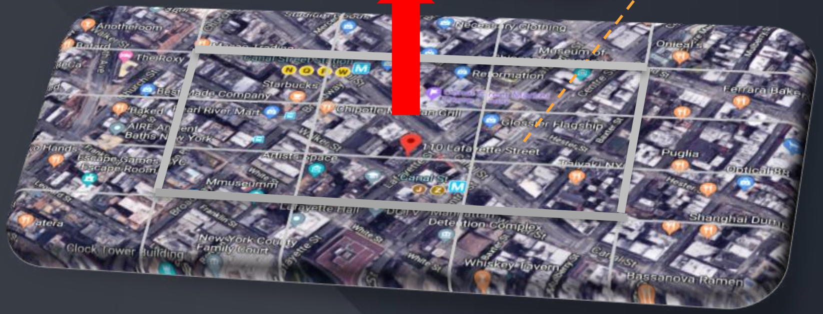
Category	Precision	Recall
restaurant	64%	62%
coffee shop	84%	59%
nightlife-entertainment	34%	56%
theatre,music,culture	50%	46%
museum	86%	78%
church / reglious place	92%	85%
hotel/motel	71%	67%
lodging	60%	54%
drugstore / pharmacy	98%	92%
hardware/house/garden	78%	78%
bookstore	81%	70%
hair and beauty	92%	80%
car repair	88%	83%
sports facility-venue	58%	72%
Average values	74%	70%

HERE Category Prediction in a long run

Text other than place name around storefront

Identified Objects around storefront

Prediction using neighboring places



Proposal @ HERE AI Summit 2019

Place Graph

POI 6
POI 4
POI 1
00 10 11
100 'Eat and Drink'
???
POI 3
POI i
200 'Going out-Entertainment'
10 10 11
01 00 11
POI 7
11 10 01
POI 2
POI 5
200 'Going out-Entertainment'

Graph Neural Network

$$\mathcal{G} = (\mathcal{A}, \mathcal{X})$$

Preprocessed Adjacency matrix $A \in R^{NxN}$

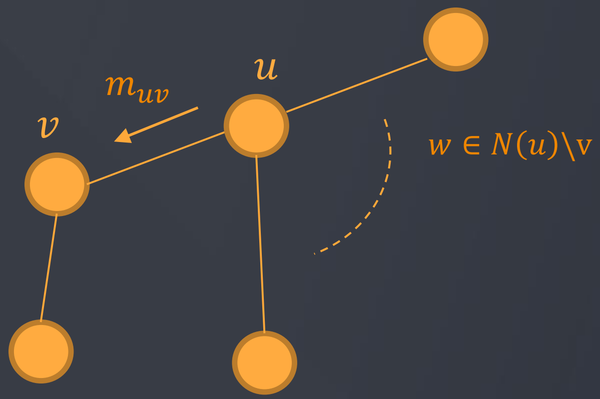
Feature matrix $X \in R^{NxF}$

$$h_v = f\,(X_v, X_{co[v]}, h_{ne[v]}, X_{ne[v]})$$
$$o_v = g\,(h_v, X_v)$$

- f : Transition function that projects inputs onto a *d*-dimensional space
- h_v : State embedding of Node (*d*-dim vector contains information of its neighborhood)
- $X_{co[v]}$: Features of the Edges connect to v
- $X_{ne[v]}$: Features of neighboring nodes of v
- $h_{ne[v]}$: Embedding of the neighboring nodes of v
- o_v : output of node v computed by state h_v and node feature x_v
- g : feed-forward fully connected neural network

GNN : Nodes and Edges embedding in the graph

Main idea: Pass **messages** between pairs of **nodes** & **agglomerate**



* Graph is bidirectional in this presentation

- Each node has feature vector : $X_v = [0, 0, 1]$
- Each edge has feature vector : $X_{uv} = [0, 1, 1]$
- Embedding of node u :

$$h_u = \tanh(U_1 X_u + U_2 * \sum_{w \in N(u)} m_{wu})$$

- Message from node u to node v :

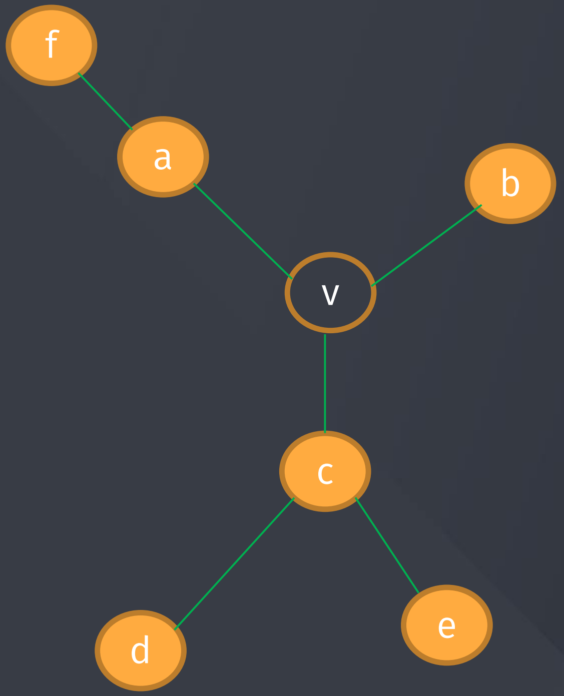
$$m_{uv} = \tanh(W_1 X_u + W_2 X_{uv} + W_3 * \sum_{w \in N(u)\backslash v} m_{wu})$$

- Iterative Neighbor Aggregation (Message passing)

simple forward NN

$$h_v^{(k+1)} = \sum_{u \in N(v)} MLP\left(h_v^{(k)}, h_u^{(k)}\right)$$

Graph Readout

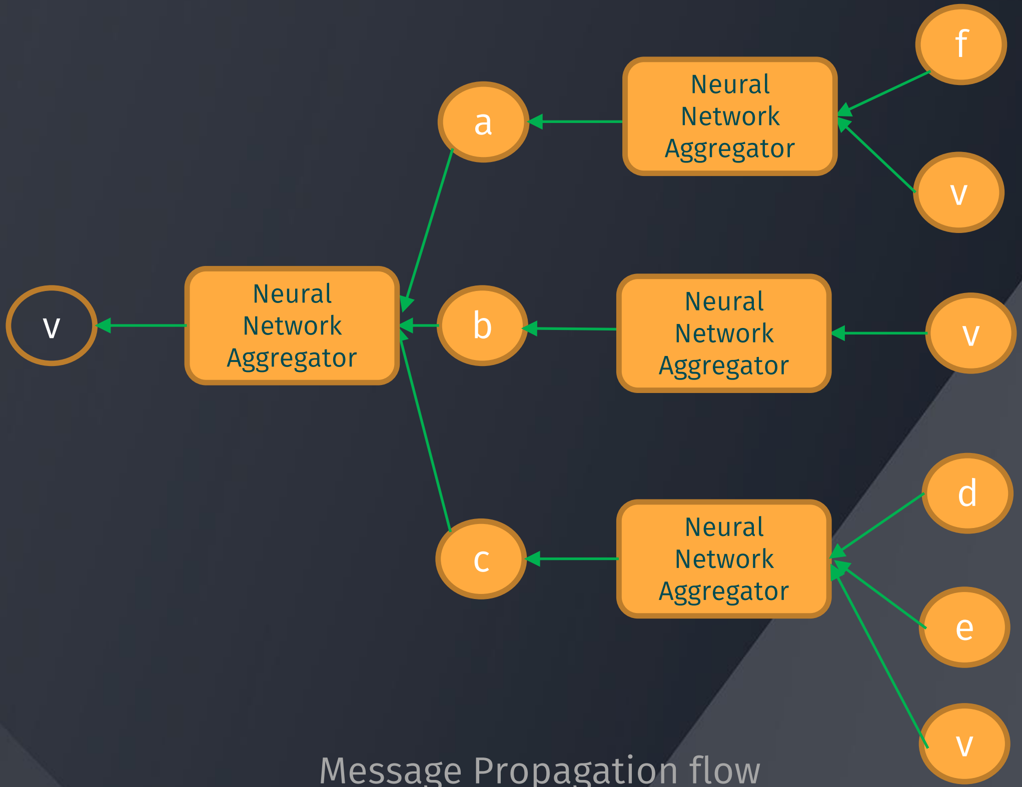


Graph structure

Neighbor Aggregation

simple forward NN

$$h_v^{(k+1)} = \sum_{u \in N(v)} MLP\left(h_v^{(k)}, h_u^{(k)}\right)$$



Message Propagation flow

Neighborhood Aggregation

Algorithm 1: Neighborhood-aggregation encoder algorithm. Adapted from [28].

Input : Graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$; input features $\{\mathbf{x}_v, \forall v \in \mathcal{V}\}$; depth K; weight matrices $\{\mathbf{W}^k, \forall k \in [1, K]\}$; non-linearity σ; differentiable aggregator functions $\{\text{AGGREGATE}_k, \forall k \in [1, K]\}$; neighborhood function $\mathcal{N} : v \rightarrow 2^{\mathcal{V}}$

Output: Vector representations $\mathbf{z}_v$ for all $v \in \mathcal{V}$

$\mathbf{h}_v^0 \leftarrow \mathbf{x}_v, \forall v \in \mathcal{V}$;

for $k = 1...K$ **do**

  for $v \in \mathcal{V}$ **do**

    $\mathbf{h}_{\mathcal{N}(v)}^k \leftarrow \text{AGGREGATE}_k(\{\mathbf{h}_u^{k-1}, \forall u \in \mathcal{N}(v)\})$;

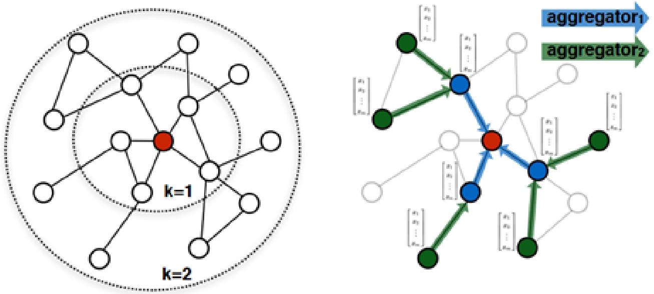
    $\mathbf{h}_v^k \leftarrow \sigma\left(\mathbf{W}^k \cdot \text{COMBINE}(\mathbf{h}_v^{k-1}, \mathbf{h}_{\mathcal{N}(v)}^k)\right)$

  end

  $\mathbf{h}_v^k \leftarrow \text{NORMALIZE}(\mathbf{h}_v^k), \forall v \in \mathcal{V}$

end

$\mathbf{z}_v \leftarrow \mathbf{h}_v^K, \forall v \in \mathcal{V}$



1. Collect neighbors

2. Aggregate feature information from neighbors

Aggregation types

- Mean aggregator

$$h_v^k \leftarrow \sigma(W \cdot mean(\{h_v^{k-1}\} \cup \{h_u^{k-1}, \forall u \in N(v)\})$$

- LSTM aggregator

  - Advantage of larger expressive capability
  - But not inherently symmetric (i.e. not permutation invariant)

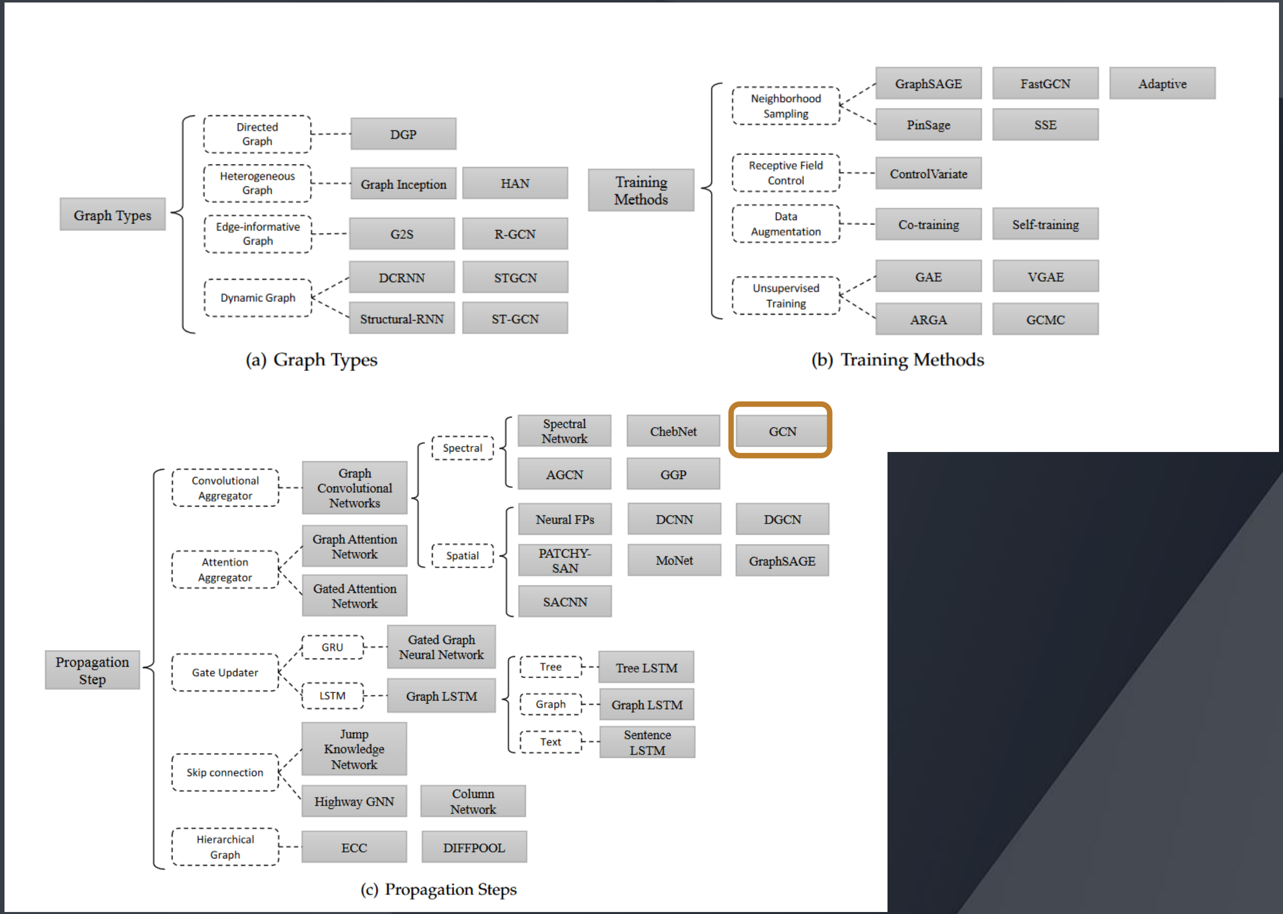
- Pooling aggregator

$$Aggregate_k^{pool} \leftarrow \max(\{\sigma(W_{pool} h_{u_i}^k + b), \forall u \in N(v)\})$$

➔ idea : differentiate the relevance of neighbor?

Variant Graph Network

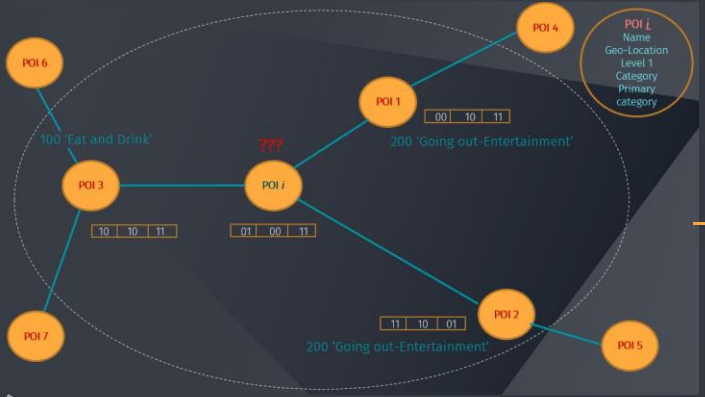
* Graph Neural Networks : A Review of Methods and Applications

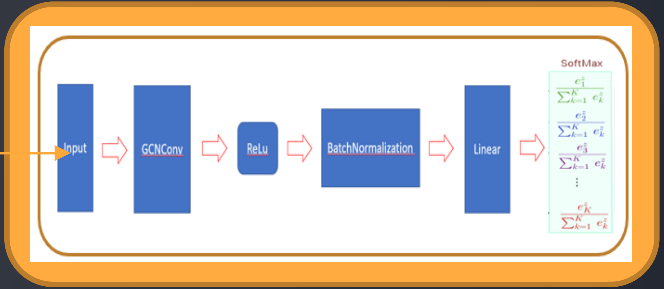


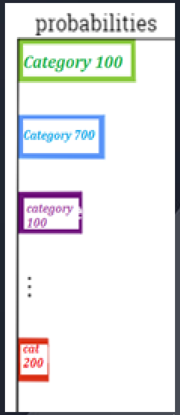
700 'Custom Services'
Label encoding [0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0]
POI 6
POI 4
POI i
Name
Geo-Location
Level 1
Category
POI 1
00 10 11
100 'Eat and Drink'
Label encoding [0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0]
???
200 'Going out-Entertainment'
Label encoding [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1]
POI 3
POI i
10 10 11
01 00 11
Input feature 'place name' in 768-dim vector by BERT
POI 2
11 10 01
POI 7
POI 5
200 'Going out-Entertainment'
Label encoding [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1]
01 01 11
100 'Eat and Drink'
here

Place Graph Embedding and Category Prediction with GCN

Category Prediction





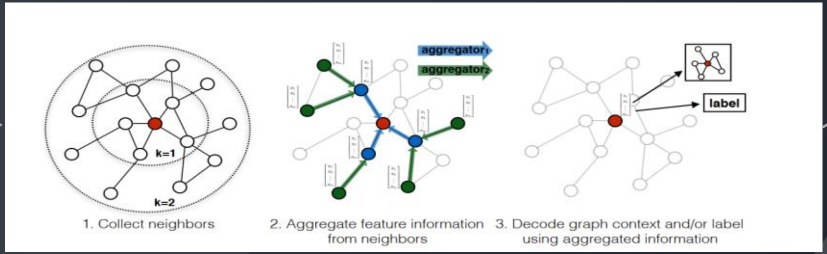


$G = (A, X)$

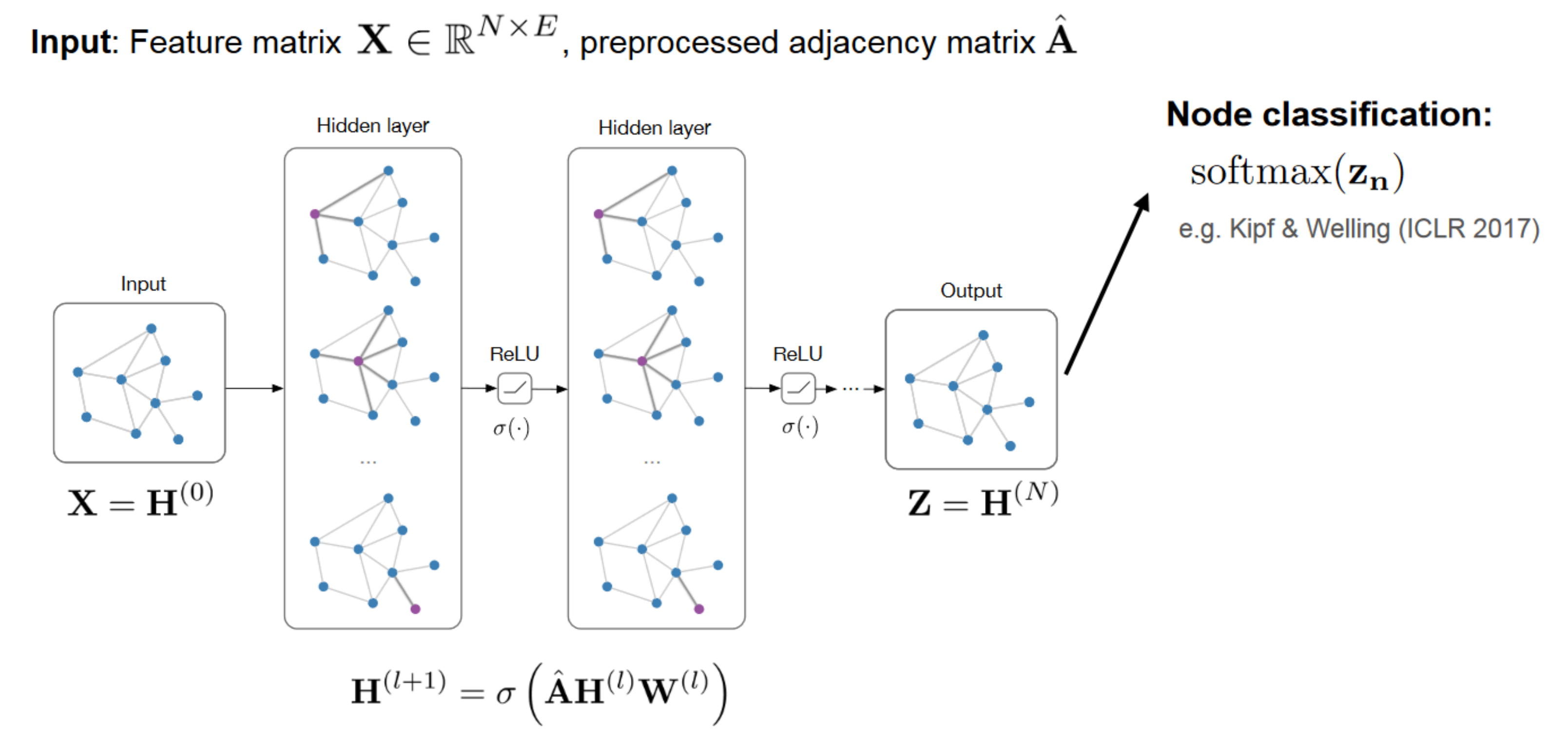
Adjacency matrix $A \in R^{NxN}$

Feature matrix $X \in R^{NxF}$

Encoder-Decoder Approach using GCN

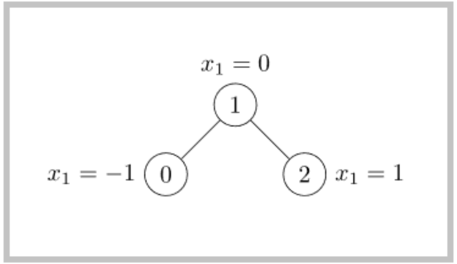


Node classification with GNN/GCN



Credit : Thomas Kipf (http://tkipf.github.io/misc/SlidesCambridge.pdf)

Graph Representation using Adjacency Matrix



1. Adjacency Matrix Represenatation

	0	1	2
0	0	1	0
1	1	0	0
2	0	1	0

Adjacency Matrix representation of Undirected Graph

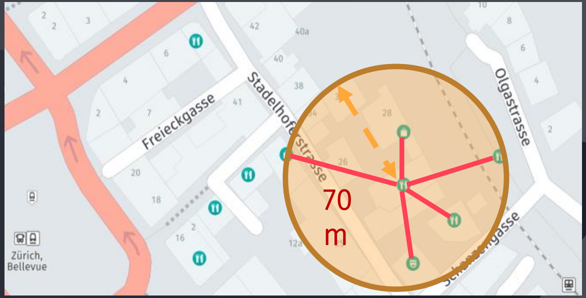
2. Develop using PyTorch GCN

```
import torch
from torch_geometric.data import Data

edge_index = torch.tensor([[0, 1, 1, 2],
                           [1, 0, 2, 1]], dtype=torch.long)
x = torch.tensor([[-1], [0], [1]], dtype=torch.float)
y = torch.tensor([[1], [0], [0]], dtype=torch.long)
dataset = Data(x=x, edge_index=edge_index, y=y)
```

New York Place Graph

- Neighboring places : Distance within 70m
- Number of nodes (places) in New York : 806,830
- Number of edges in New York graph : 4,033,104



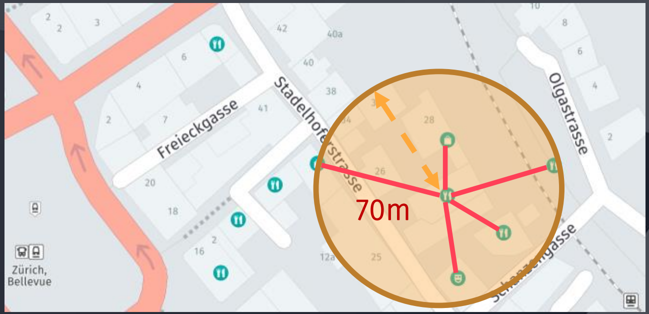
data	size	elapsed time to create
New York place data (806,830 places)	119 MB	
New York data place name and label embedding generation (using BERT)	10.2 GB	35 hour in 125 GB memory GPU machine
New York Adjacency matrix size	86.3 MB	9 hour to generate
Number of edges in adjmatrix	4,033,105 (4 million edges)	

Place data analysis on New York and Chicago

city	number of places	bounding box
New York	806,830	{'left_lon':-74.2591, 'bottom_lat':40.4774, 'right_lon':-73.7002, 'top_lat':40.9162}
Chicago	248,024	{'left_lon':-88.133, 'bottom_lat':41.6062, 'right_lon':-87.4656, 'top_lat':42.1603}

Chicago Place Graph

- Neighboring places : Distance within 70m
- Number of nodes (places) in Chicago : 248,024
- Number of edges in Chicago graph : 2,442,558



The number of places in Chicago is 248,024 and each place contains pid, geo-location.

The data frame that contains pid, place name, level 1 category, full category, category name, place name vector (with 768 dimension), Label encoding

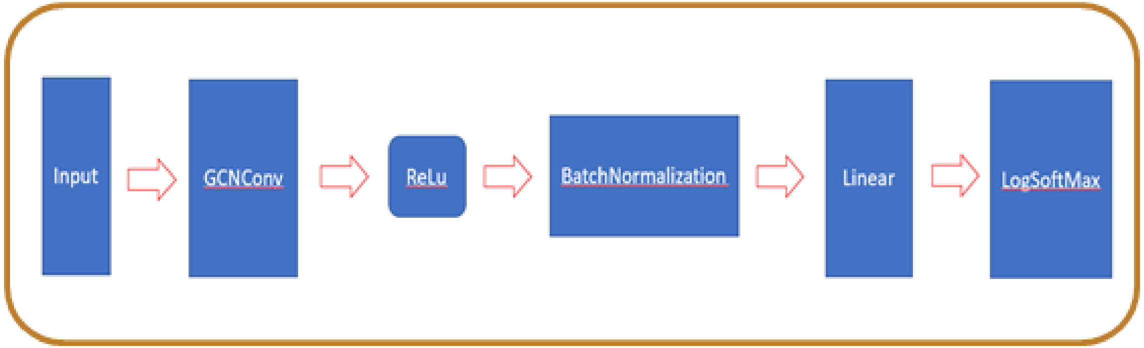
data	size	time to create
Chicago place data (248,024 places)	18.42 MB	
Chicago place adjacency matrix	50.8 MB	~ 4 hours in GPU machine (with 125 GB memory)
Number of edges (connections between places)	2,442,558 (2 million edges)	
Chicago dataframe contains place name vectors using BERT	3.13 GB	~ 8 hours to generate dataframe
NYC dataframe contains place name vectors using BERT	10.2 GB	~ 1 day to generate dataframe

Graph Convolutional Model Architecture

```
class Net(torch.nn.Module):
    def __init__(self):
        super(Net7, self).__init__()

        self.conv1 = GCNConv(dataset.num_node_features, 256)
        self.conv2 = GCNConv(126, 62)
        self.linear1 = torch.nn.Linear(256, 11)
        self.linear2 = torch.nn.Linear(126, 11)
        self.pool1 = TopKPooling(512, ratio=0.5)
        self.pool2 = TopKPooling(62, ratio=0.5)
        self.bn1 = torch.nn.BatchNorm1d(256)
        self.bn2 = torch.nn.BatchNorm1d(62)


    def forward(self, data):
        x, edge_index = data.x, data.edge_index
        x = self.conv1(x, edge_index)
        x = F.relu(x)
        x = self.bn1(x)
        x = self.linear1(x)
        return F.log_softmax(x, dim=1)
```

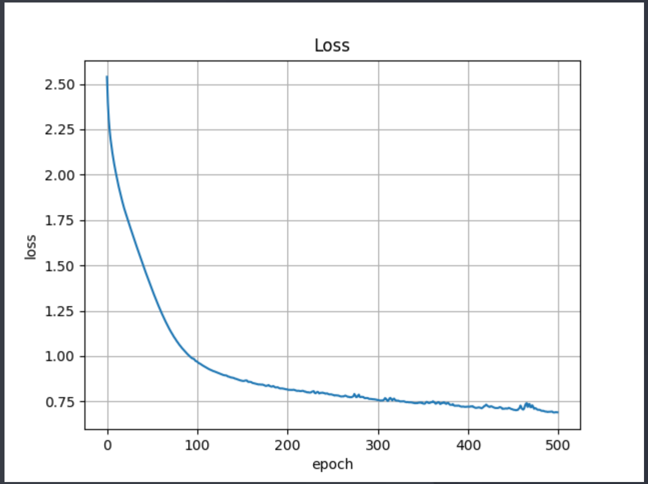


Model Architecture - Forward Pass

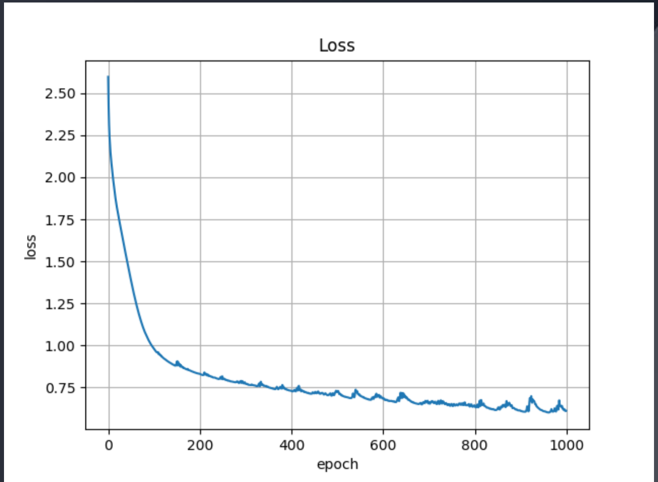
GCNConv : The graph convolutional operator from the *"Semi-supervised Classification with Graph Convolutional Networks"*

Chicago Graph Model Accuracy

- Number of Places : 248,024
- Neighboring distance : 70m
- Number of edges in graph : 2,442,558 (2.4 M edges)
- Learning Rate : 0.001
- Optimizer : Adam
- Loss function : Cross Entropy Loss



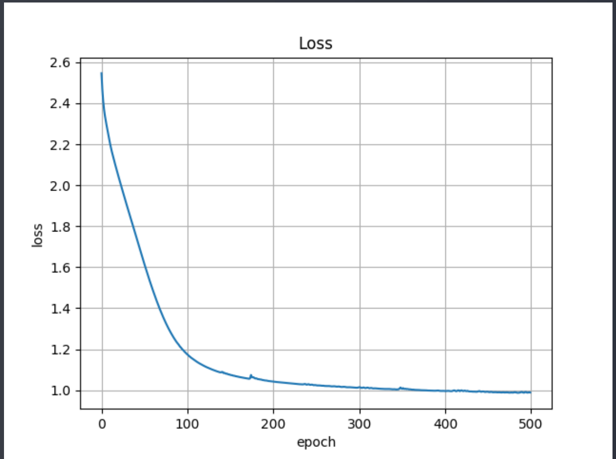
Epoch 500
Accuracy : 0.75



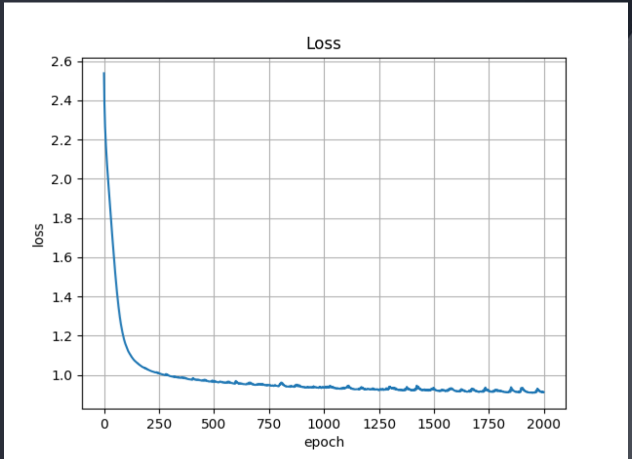
Epoch 1000
Accuracy : 0.7846

New York Graph Model Accuracy

- Number of Places : 806,830
- Neighboring distance : 70m
- Number of edges in graph : 4,033,105 (4.03 M edges)
- Learning Rate : 0.001
- Optimizer : Adam
- Loss function : Cross Entropy Loss



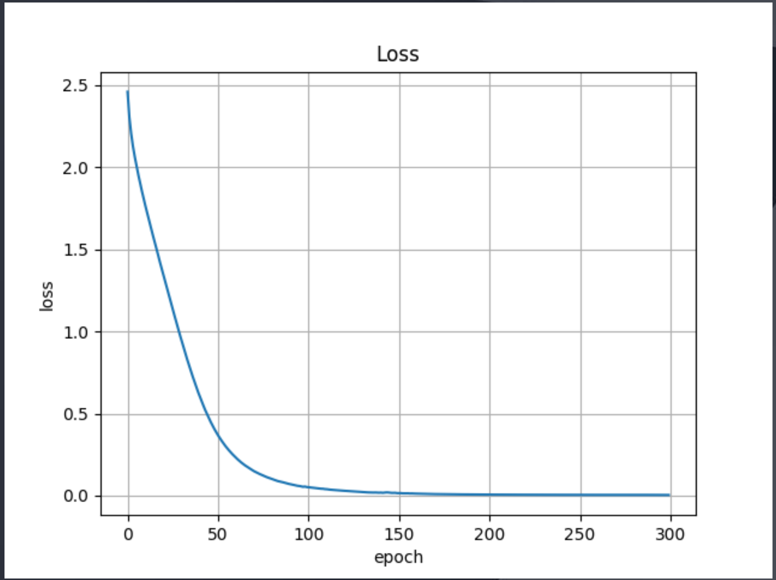
Epoch 500
Accuracy : 0.63



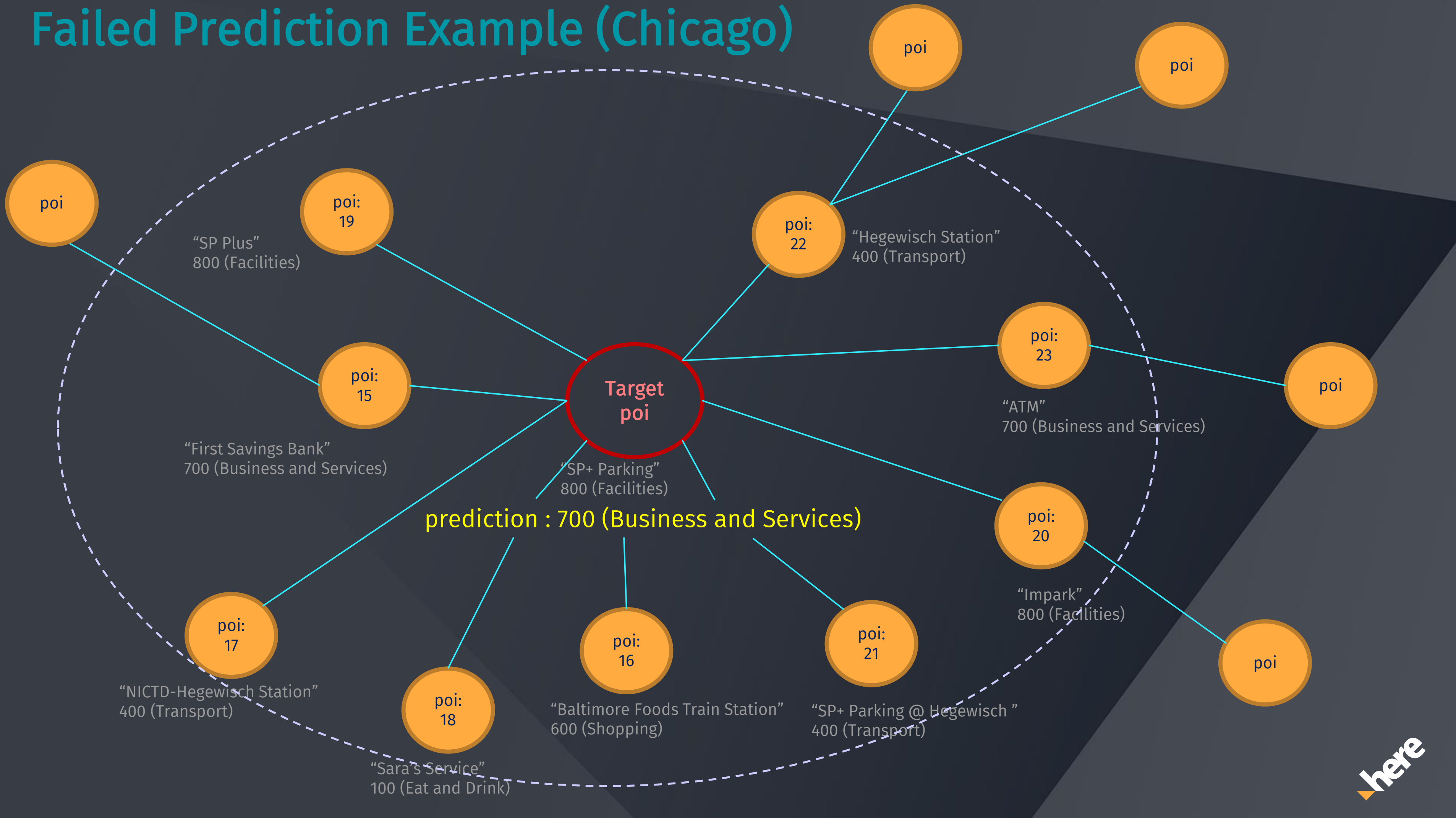
Epoch 2000
Accuracy : 0.67

New York (very small) Graph Model Accuracy

- Number of places : 1000
- Neighboring distance : 50m
- Accuracy : **0.87**
- Epochs : 300
- Learning Rate : 0.001
- Optimizer : Adam
- Loss function : Cross Entropy Loss

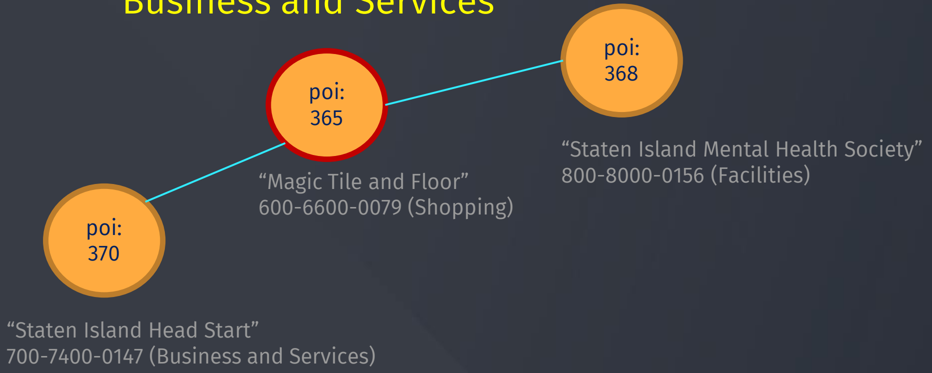


Failed Prediction Example (Chicago)

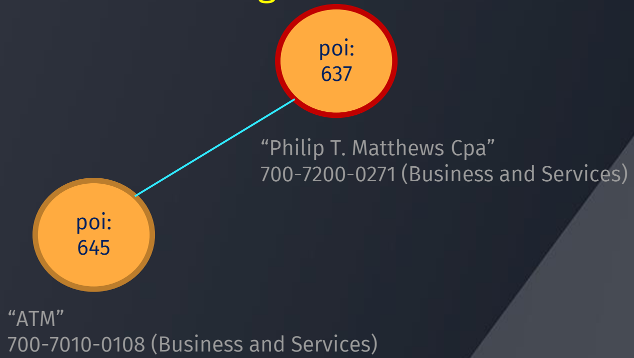


Failed Prediction Example (New York)

prediction : 700
Business and Services



prediction : 200
Going out - Entertainment



Open problems

In general

- Scalability : Graph can be extremely big!
- Interpretability

In HERE place prediction

- Effect of different Aggregator
- Graph size matters for better prediction?
- Encoding with probability of category distribution