8월 7일

오늘 교수님께 메일을 보내야한다.

1. relation을 text로 변환하기 위해서 어떤 text로 할 지 정해야함

Relation to text

각 relation앞에는 A, 뒤에는 B가 붙어서 해석하면 편합니다.

EX) “A(samsung) is at location of B(seoul)” ↔ “B(seoul) is considered as the location of A(samsung)”

0,19 : context to question | “is context and that is question “ | “is question and that is context”
1,20 : context to answer | “is context and that is answer” | “is answer and that is context”
2,21 : antonym | “is the antonym of” | “is the antonym of”
3,22 : atlocation | “is at location of “ | “ is considered as the location of “
4,23 : capableof | “is capable of” | “ is capable of being”
5,24 : causes | “causes” | “is caused by”
6,25 : createdby | “is created by” | “creates”
7,26 : isa | “is a” | “is a”| “is a”
8,27 : desires | “desires” | “desires”
9,28 : hassubevent | “has subevent” | “is the subevent of “
10,29 : partof | “is part of” | “includes”
11,30 : hascontext | “has context” | “is the context of “
12,31 : hasproperty | “has property” | “is the property of”
13,32 : madeof | “is made of” | “is the material of”
14,33 : notcapableof | “is not capable of” | “is not capable of being”
15,34 : notdesires | “does not desire” | “does not desire”
16,35 : receivesaction | “is” | “is”
17,36 : relatedto | “is related to” | “is related to“
18,37 : usedfor | “is used for” | uses

Relation to text (1)

```
import torch
import time
# Define the edge_index and edge_type arrays
edge_index = torch.tensor([[3, 2, 2, 9, 9, 9, 10, 3, 1, 1, 2, 2, 2, 2, 2, 3, 5,
                            6, 7, 8, 8, 8, 8, 9, 9, 0, 0, 0, 8, 5, 7, 5, 7, 1, 3,
                            4, 6, 7, 4, 5, 6, 7, 9, 10, 2, 1, 1, 10, 1, 7, 9, 10, 1,
                            6, 1, 2, 3],
                           [8, 5, 7, 5, 7, 1, 3, 4, 6, 7, 4, 5, 6, 7, 9, 10, 2, 1,
                            1, 10, 1, 7, 9, 10, 1, 6, 1, 2, 3, 3, 2, 2, 9, 9, 9, 10,
                            3, 1, 1, 2, 2, 2, 2, 2, 2, 3, 5, 6, 7, 8, 8, 8, 8, 9,
                            9, 0, 0, 0]])
edge_type = torch.tensor([4, 7, 7, 7, 7, 9, 10, 12, 17, 17, 17, 17, 17, 17, 17, 17, 17, 17,
                          17, 17, 17, 17, 17, 17, 17, 17, 0, 0, 0, 23, 26, 26, 26, 26, 28, 29,
                          31, 36, 36, 36, 36, 36, 36, 36, 36, 36, 36, 36, 36, 36, 36, 36, 36,
                          36, 19, 19, 19])

# Get the number of nodes
num_nodes = max(edge_index.max().item(), edge_index.max().item()) + 1

# Create the adjacency list representation of the graph
adj_list = {i: [] for i in range(num_nodes)}
for i, j in edge_index.T.tolist():
    adj_list[i].append(j)

# Function to find cycles of length 3 using DFS
def find_cycles_dfs(node, start_node, depth, visited, path, cycles):
    visited[node] = True
    path.append(node)

    if depth == 2:
        if start_node in adj_list[node]:  # Check if we can go back to the start_node
            cycle = list(path)  # Copy the path to preserve the original order
            min_idx = cycle.index(min(cycle))
```

```
            cycle = cycle[min_idx:] + cycle[:min_idx]  # Rotate to get the minimum representation
            cycles.add(tuple(cycle))
    elif depth < 2:
        for neighbor in adj_list[node]:
            if not visited[neighbor]:
                find_cycles_dfs(neighbor, start_node, depth + 1, visited, path, cycles)

    path.pop()
    visited[node] = False
a = time.time()
# Find the cycles of length 3
cycles = set()
for start_node in range(num_nodes):
    visited = [False] * num_nodes
    find_cycles_dfs(start_node, start_node, 0, visited, [], cycles)

# Convert sets back to lists
cycles = [list(cycle) for cycle in cycles]

b=time.time()
print(b-a)
print(len(cycles))
print(cycles)
```

위 코드는 edge_index와 edge_type가 한 문제에 대해서 있을 때를 나타낸것이다.

한 문제에 대해서 5개 sub graph가 나오므로 5개의 edge_index와 edge_type가 한 리스트에 들어있다.

이것을 MultiGPUSparseAdjData 클래스에 넣어야한다.

우선, 위 코드의 결과 cycles는 사이클에 속하는 노드들의 리스트들이다.

```
[[0, 2, 3], [0, 3, 2], [1, 5, 9], [1, 6, 9], [1, 7, 8], [1, 7, 9], [1, 8, 7], [1, 8, 9], [1, 9, 5], [1, 9, 6], [1, 9, 7], [1, 9, 8], [
```

[0, 2, 3]이 의미하는 것은 0→2 에지, 2→3, 3→0에지로 구성된 순환 그래프를 나타낸다.

0→2 에지의 타입은 0, 2→3 에지의 타입은 36, 3→0 에지의 타입은 19이다.

기존의 cycles리스트로부터 사이클을 형성하는 에지타입 코드도 완성했지만 cycles을 찾는 코드와 cycles에 해당하는 edge type을 찾는 코드를 각각 따로 하는 경우는 비효율적이다.(시간복잡도 증가)

그러므로 cycles을 찾는 과정에서 에지를 보기 때문에 바로 할 수 있는 코드를 만들어보자.

예를 들어, [0,2,3]이 아니라, [0,0,2,2,36,3,3,19,0]형태로 나올 수 있도록 말이다.

잠깐!!!

MultiGPUSparseAdjData에서 할지말고

LM_QAGNN_DataLoader 클래스에서 adj데이터를 불러오니까 거기서 바로 해버리면 되겠다.

```
num_choice = self.train_encoder_data[0].size(1)
self.num_choice = num_choice
print ('num_choice', num_choice)
*self.train_decoder_data, self.train_adj_data = load_sparse_adj_data_with_contextnode(train_adj_path, max_node_num, num_choice, args)

print('출력출력출력',len(self.train_adj_data[0]))
assert -1 == 0
*self.dev_decoder_data, self.dev_adj_data = load_sparse_adj_data_with_contextnode(dev_adj_path, max_node_num, num_choice, args)
```

load_sparse_adj_data_with_contextnode 함수로부터 나온 self.train_adj_data를 가지고 하면 될듯

이 데이터엔 총 학습데이터 9741문제가 들어있다.

9741 * 5 = 48705개

내일은 이것(총 9741개의 문제)을 기반으로 코드를 만들어보자