9월 18일

왜인지 모르겠으나.. 1시까지 헤롱헤롱 정신을 못참

seed값을 변경해서 돌려보고 노드 개수 64개 및 세이프디 과제를 진행해보자

세이프디 과제 참고 자료 :https://bong-sik.tistory.com/30

```python
# convert Cycle Triple to Cycle Passage
def convert_cycle_to_text(concept_ids, cycle_triple,qa_context): # cycle_triple에는 5개의 subgraph(1-1,1-2,1-3,1-4,1-5)가 들어감
    with open('./data/cpnet/concept.txt', "r", encoding="utf8") as fin:
        id2concept = [w.strip() for w in fin]
        # print(id2concept[-1])
    relation_text = [
    'is context and that is question',
    'is context and that is answer',
    'is the antonym of',
    'is at location of',
    'can', # can 이 어떰?
    'causes',
    'is created by',
    'is a',
    'desires',
    'has subevent',
    'is part of',
    'has context',
    'has property',
    'is made of',
    'cannot', # cannot이 어떰
    'does not desires',
    'is',
    'is related to',
    'is used for',
    'is question and that is context',
    'is answer and that is context',
    'is the antonym of',
    'is considered as the location of',
    'can be done by', #  A is something that B can do
    'is caused by',
    'creates',
    'is a',
    'desires',
    'is the subevent of',
    'includes',
    'is the context of',
    'is the property of',
    'is used to make',
    'cannot be done by', # A is something that B cannot do
    'does not desires',
    'is',
    'is related to',
    'uses'
        ]

    # Function to convert a cycle triple to text
    def convert_cycle_to_text_instance(cycle_instance,qa_context):
        text_cycle_instance = []
        for head, edge, tail in cycle_instance:
            # if not head and not edge and not tail: # 사이클이 없는 문제 pass
            #     text_cycle_instance.append([])
            #     continue
            # print('concept_ids',concept_ids)
            # print('head',head)
            # print('edge',edge)
            # print('tail',tail)
            if head == 0:
                head_text = qa_context
            else:
                head_text = id2concept[concept_ids[head]-1]
            edge_text = relation_text[edge]

            if tail == 0:
                tail_text = qa_context
            else:
                tail_text = id2concept[concept_ids[tail]-1]


            # if edge_text in ("can be done by", "cannot be done by"):
            #     head_text += 'ing'
            # elif edge_text == 'is used to make' and head_text in ("human","person"):
            #     edge_text = 'is used to form'
            # elif edge_text == 'is context and that is question':
            #     text_cycle_instance.append(f'{head_text} is context and {tail_text} is question')
            # elif edge_text == 'is context and that is answer':
            #     text_cycle_instance.append(f'{head_text} is context and {tail_text} is answer')
            # elif edge_text == 'is question and that is context':
            #     text_cycle_instance.append(f'{head_text} is question and {tail_text} is context')
            # elif edge_text == 'is answer and that is context':
            #     text_cycle_instance.append(f'{head_text} is answer and {tail_text} is context')


            # text_cycle_instance.append(f"{head_text} {edge_text} {tail_text}")
            if edge_text == 'is context and that is question':
                text_cycle_instance.append(f'{head_text} is context and {tail_text} is question')
            elif edge_text == 'is context and that is answer':
                text_cycle_instance.append(f'{head_text} is context and {tail_text} is answer')
            elif edge_text == 'is question and that is context':
                text_cycle_instance.append(f'{head_text} is question and {tail_text} is context')
            elif edge_text == 'is answer and that is context':
                text_cycle_instance.append(f'{head_text} is answer and {tail_text} is context')
            elif edge_text in ("can be done by", "cannot be done by"):
                head_text += 'ing'
                text_cycle_instance.append(f"{head_text} {edge_text} {tail_text}")
            elif edge_text == 'is used to make' and head_text in ("human","person"):
                edge_text = 'is used to form'
                text_cycle_instance.append(f"{head_text} {edge_text} {tail_text}")
            # Add other edge type conditions here
            else:
                text_cycle_instance.append(f"{head_text} {edge_text} {tail_text}")


        # Combine the three sentences within the cycle_instance into one sentence. It is used for Documents
        combined_sentence = [" and ".join(text_cycle_instance)]
        return combined_sentence

    # Convert cycle_triple to text representation
    text_representation = []

    for cycle_instance in cycle_triple: # 1-1

        text_cycle_instance = convert_cycle_to_text_instance(cycle_instance,qa_context)
        text_representation.append(text_cycle_instance)

    return text_representation


import time
def convert_nested_cycles_to_text(concept_ids, cycle_triple_list,qa_context): # qa_context = 9741
    text_representation_list = []

    for i in range(len(cycle_triple_list)):  # Loop over the problems (9741)
        problem_text_representation = []  # Store text representation for each problem's answer options

        for j in range(len(cycle_triple_list[i])):  # Loop over each answer option (5)
            concept_ids_graph = concept_ids[i * 5 + j]  # Extract concept IDs for one graph
            cycle_triples_graph = cycle_triple_list[i][j]  # Extract cycle triples for one graph

            if not cycle_triples_graph: # subgraph에 사이클이 없을 경우
                problem_text_representation.append([])
                continue
            # Convert one graph to text representation
            # print(concept_ids_graph)
            # print(cycle_triples_graph)
            graph_text_representation = convert_cycle_to_text(concept_ids_graph, cycle_triples_graph,qa_context[i][j])
            # print('graph',graph_text_representation)
            # assert -1 == 0
            problem_text_representation.append(graph_text_representation)
            # print('j처리')
        text_representation_list.append(problem_text_representation)

        print('i처리')
    return text_representation_list


a=time.time()
text_representations = convert_nested_cycles_to_text(train_concept_ids, train_cycle_triple,train_qa_context)
print(time.time()-a)
```

위 코드가 cycle_triple을 cycle_passage로 바꾸는 코드인데 이것부터가 문제가 있었다.

기존에는

```
if edge_text in ("can be done by", "cannot be done by"):
                head_text += 'ing'
elif edge_text == 'is used to make' and head_text in ("human","person"):
                edge_text = 'is used to form'
elif edge_text == 'is context and that is question':
                text_cycle_instance.append(f'{head_text} is context and {tail_text} is question')
elif edge_text == 'is context and that is answer':
                text_cycle_instance.append(f'{head_text} is context and {tail_text} is answer')
elif edge_text == 'is question and that is context':
                text_cycle_instance.append(f'{head_text} is question and {tail_text} is context')
elif edge_text == 'is answer and that is context':
                text_cycle_instance.append(f'{head_text} is answer and {tail_text} is context')
text_cycle_instance.append(f"{head_text} {edge_text} {tail_text}")
```

였는데 이는 제대로 된 구분을 못해주었다.

그래서 고친 것이 아래와 같다.

```
if edge_text == 'is context and that is question':
                text_cycle_instance.append(f'{head_text} is context and {tail_text} is question')
elif edge_text == 'is context and that is answer':
                text_cycle_instance.append(f'{head_text} is context and {tail_text} is answer')
elif edge_text == 'is question and that is context':
                text_cycle_instance.append(f'{head_text} is question and {tail_text} is context')
elif edge_text == 'is answer and that is context':
                text_cycle_instance.append(f'{head_text} is answer and {tail_text} is context')
elif edge_text in ("can be done by", "cannot be done by"):
                head_text += 'ing'
                text_cycle_instance.append(f"{head_text} {edge_text} {tail_text}")
elif edge_text == 'is used to make' and head_text in ("human","person"):
                edge_text = 'is used to form'
                text_cycle_instance.append(f"{head_text} {edge_text} {tail_text}")
```

그래서 현재 9시 40분까지 열심히 context score 를 만드는 코드를 수정했지만 사실 다 처음부터 해야한다는 사실…

그래서 생각한 순서!!

1. 일단, 노드 32개를 대상으로 하는 전처리 과정을 colab에서 진행한다. (자기전에 train_data 대상으로 BM25 돌리고 자기, train데이터도 cycle_passage는 다시 만들어야함
2. 노드 64개를 대상으로 하는 과정은 cycle_triple,edge_index,edge_type,qa_context,concept_ids을 만드는 과정까지만 서버에서 진행한다. + cycle_passage (오늘까지 진행)

데이터 형태 한 번 정리하자

edge_index : 48705

edge_type : 48705

cycle_triple : 9741

cycle_passage : 9741

qa_context : 9741

dpr : 48705

bm25 :

내일 진행할 사항 정리하기

1. 세이프디 최종 정리하기 → 1순위
2. 노드 32개 전처리 진행 완료 및 new context score 제작 완료하기
3. 노드 64개 전처리 완료하기