Exercise#1: Hands-on Practice of an Inductive KGRL Method

Joyce Jiyoung Whang
School of Computing, KAIST

Key Facets in Modern Knowledge Graph Representation Learning (KeyKGRL), ISWC 2025 Tutorial

https://bdi-lab.kaist.ac.kr

KAIST BDILab
BIG DATA INTELLIGENCE

01 Overview

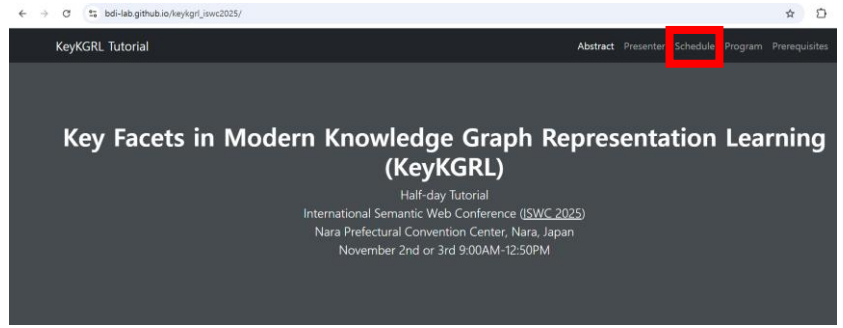
- We run an inductive KGRL method, InGram, on Google Colab.
  - We use FB-100 in this exercise

- **Part #1: Inspecting a relation graph**
  - We will create a relation graph
  - We choose a relation and inspect which relations have high affinity scores and which relations have low affinity scores with the chosen relation

- **Part #2: Reproduce the results of InGram**

02 Accessing the Exercise Material

- Method #1: Use the homepage of our Tutorial
  - 1. Access https://bdi-lab.github.io/keykgrl_iswc2025/
  - 2. Click “Schedule” at the right side of the bar on the top
  - 3. Click “[Exercise1]Hands-on Practice of Inductive KGRL” in the table

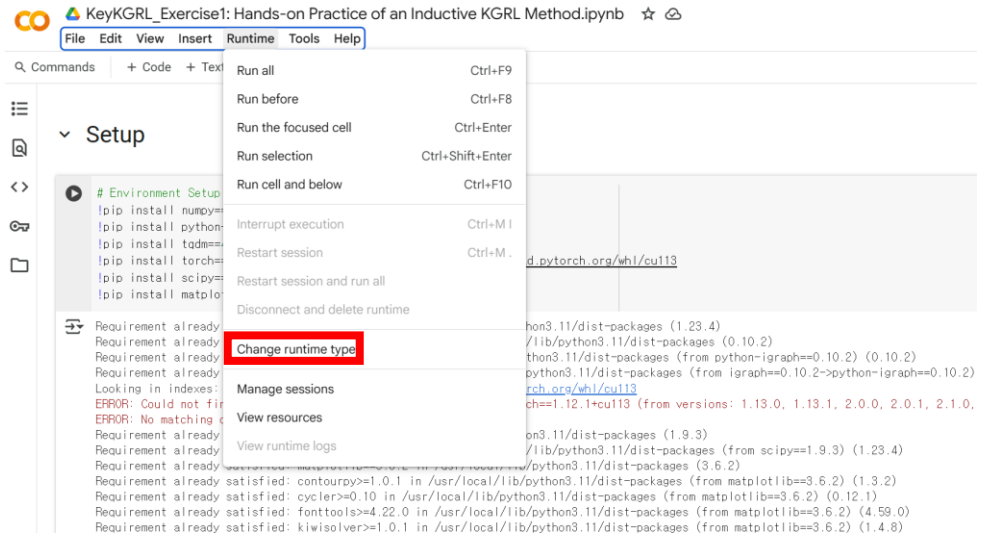


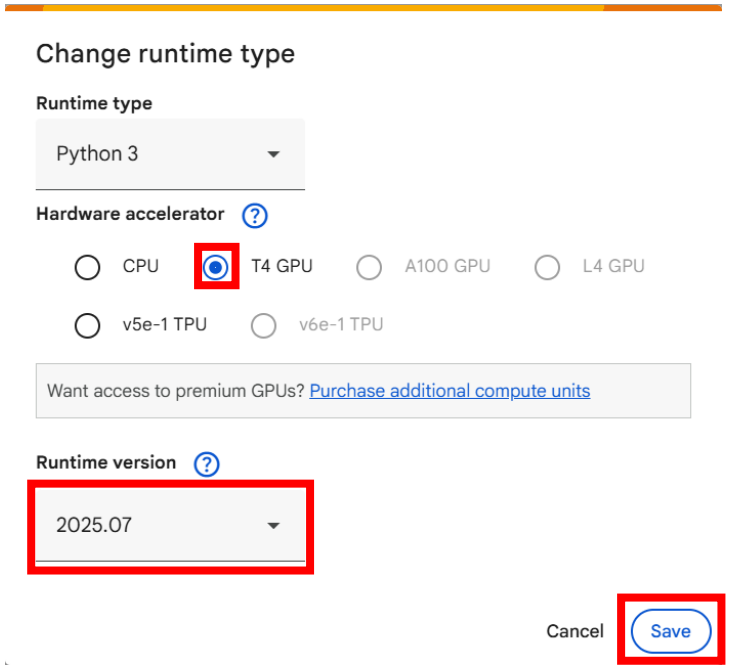
Time Slot	Tutorial Time	Program
9:00 - 9:40	9:00-9:10	Opening & Introduction to Knowledge Graphs
	9:10-9:45	[Lecture 1] KG Embedding with Multimodal Data
	9:45-10:20	[Lecture 2] Inductive Reasoning on KGs
	10:20-10:40	[Exercise 1] Hands-on Practice of Inductive KGRL
10:40-11:10	Break Time	

- Method #2: Direct Link
  - https://colab.research.google.com/drive/1abGO3o56pbuQA0WzfaLFNh0hWEyelgQW?usp=sharing

02 Environment Setup

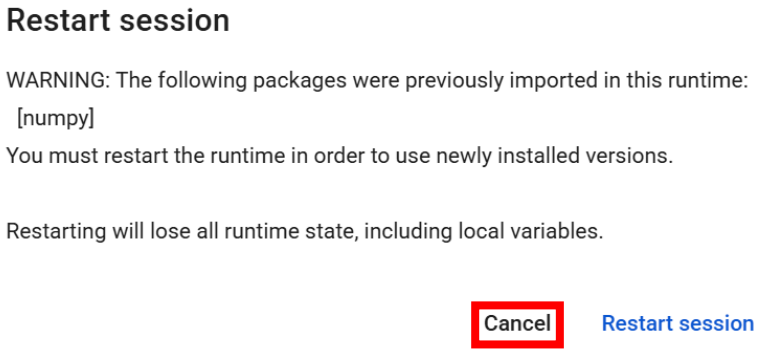
- We use a GPU in this exercise
  - Runtime -> Change runtime type -> click a GPU in hardware accelerator -> save
- We use previous runtime version
  - Runtime -> Change runtime type -> Runtime version -> 2025.07



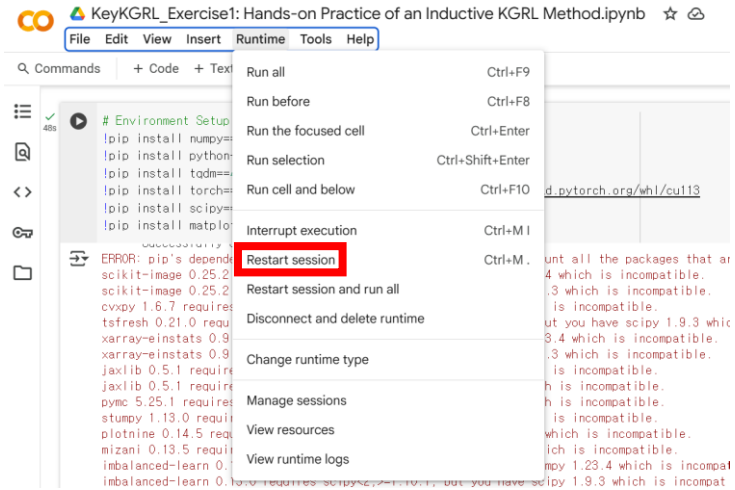


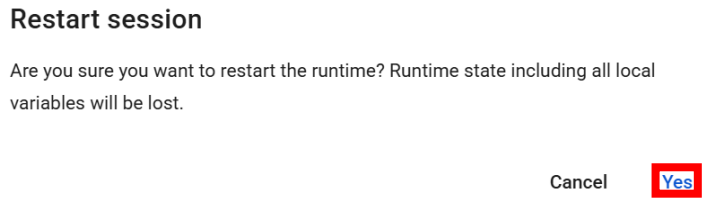
02 Environment Setup

- 1. Run the first cell, and wait until it finishes running
  - Ignore the warning and click “Cancel”



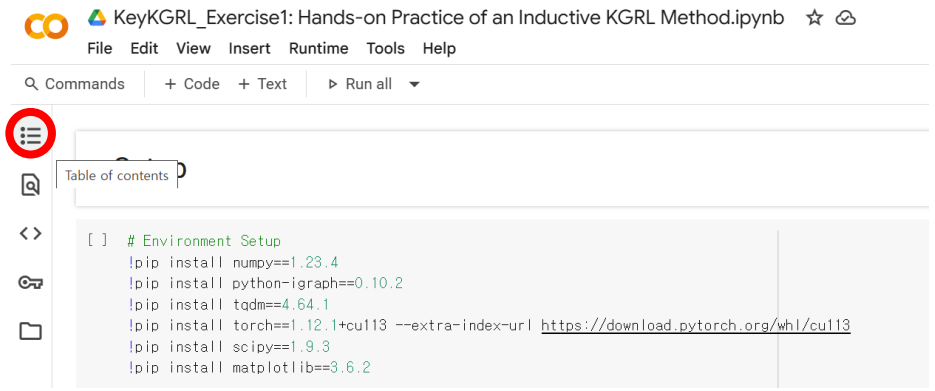
- 2. Restart the session after the cell is finished



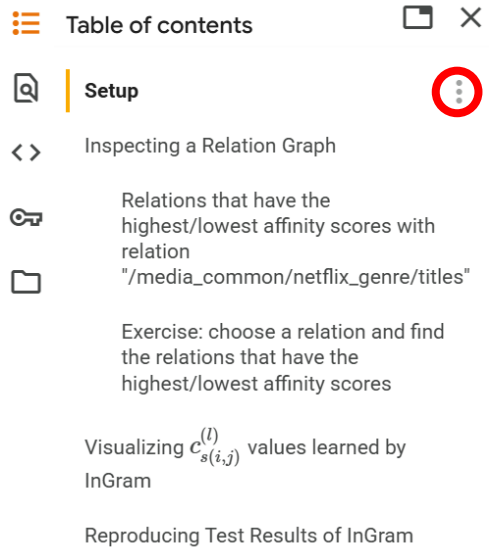


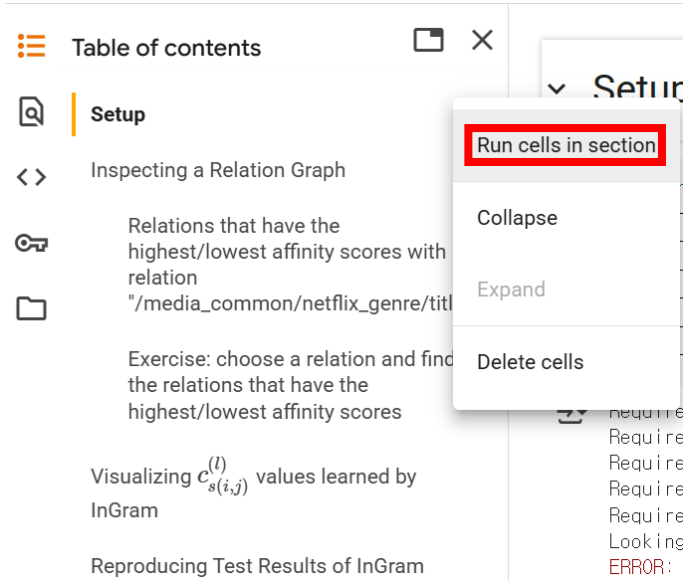
02 Environment Setup

- 3. Open the table of contents in the left side bar



- 4. Run all cells in the section “Setup”





03 Inspecting a Relation Graph

- We create a relation graph using the provided code

```
[ ]  # Create a Relation Graph
     test_relation_triplets = generate_relation_triplets(test.msg_triplets, test.num_ent, test.num_rel, configs['num_bin'])
```

- First, we analyze the relation “/media_common/netflix_genre/titles”
  - Examine which relations have high affinity scores and which relations have low scores

```
[ ]  target_id = test.rel2id['/media_common/netflix_genre/titles']


     print("==========Relations that have the highest affinity scores with ₩"/media_common/netflix_genre/titles₩"==========")
     for r1,r2,s12 in test_relation_triplets:
       if r1 == target_id and s12 == 0:
         print(test.id2rel[r2%test.num_rel]+ ("_inv" if r2>=test.num_rel else ""))
```

```
[ ]  print("==========Relations that have the lowest affinity scores with ₩"/media_common/netflix_genre/titles₩"==========")
     for r1,r2,s12 in test_relation_triplets:
       if r1 == target_id and s12 == configs['num_bin']-1:
         print(test.id2rel[r2%test.num_rel]+ ("_inv" if r2>=test.num_rel else ""))
```

03 Inspecting a Relation Graph

```
==========Relations that have the highest affinity scores with "/media_common/netflix_genre/titles"==========
/media_common/netflix_genre/titles
/film/film/release_date_s./film/film_regional_release_date/film_release_distribution_medium_inv
/film/film/other_crew./film/film_crew_gig/film_crew_role_inv
/film/film/genre_inv


==========Relations that have the lowest affinity scores with "/media_common/netflix_genre/titles"==========
/sports/sports_team_location/teams
/location/statistical_region/religions./location/religion_percentage/religion
/base/aareas/schema/administrative_area/administrative_parent
/location/statistical_region/places_exported_to./location/imports_and_exports/exported_to
/military/military_combatant/military_conflicts./military/military_combatant_group/combatants
/location/country/capital
/olympics/olympic_participating_country/athletes./olympics/olympic_athlete_affiliation/olympics
/location/country/form_of_government
/base/aareas/schema/administrative_area/administrative_area_type
/people/person/places_lived./people/place_lived/location_inv
/people/person/spouse_s./people/marriage/location_of_ceremony_inv
/base/culturalevent/event/entity_involved_inv
/location/statistical_region/religions./location/religion_percentage/religion_inv
/base/aareas/schema/administrative_area/administrative_parent_inv
/people/ethnicity/geographic_distribution_inv
/location/statistical_region/places_exported_to./location/imports_and_exports/exported_to_inv
/military/military_combatant/military_conflicts./military/military_combatant_group/combatants_inv
/base/biblioness/bibs_location/country_inv
```

03 Exercise: Analyze a Relation

- Next, choose a relation you would like to analyze

- The full list of relations can be obtained by running the following cell

```
[ ]  ## List of relations
     print("===========List of Relations==========")
     for rel_name in test.id2rel:
       print(rel_name)
       print(rel_name+"_inv")
```

03 Exercise: Analyze a Relation

- Set *target_rel* as the relation you have chosen

```
[ ]  ## Choose a relation
     target_rel = ""

     #target_rel = "/media_common/netflix_genre/titles_inv"
```

- Observe the relations that have high/low affinity scores with the chosen relation

```
[ ]  print(f"===========Relations that have the highest affinity scores with ₩"{target_rel}₩"==========")
     if "_inv"!= target_rel[-4:]:
       target_id = test.rel2id[target_rel]
     else:
       target_id = test.rel2id[target_rel[:-4]]+test.num_rel
     for r1,r2,s12 in test_relation_triplets:
       if r1 == target_id and s12 == 0:
         print(test.id2rel[r2%test.num_rel]+ ("_inv" if r2>=test.num_rel else ""))
```

```
[ ]  print(f"===========Relations that have the lowest affinity scores with ₩"{target_rel}₩"==========")
     for r1,r2,s12 in test_relation_triplets:
       if r1 == target_id and s12 == configs['num_bin']-1:
         print(test.id2rel[r2%test.num_rel]+ ("_inv" if r2>=test.num_rel else ""))
```

03 Reproducing the Results of InGram

- Finally, we reproduce MR, MRR, Hit@10, and Hit@1 values of InGram on the test set of FB-100

```
[ ]  with torch.no_grad():
        my_model.eval()
        msg = torch.tensor(test.msg_triplets).cuda()
        sup = torch.tensor(test.sup_triplets).cuda()
        relation_triplets = torch.tensor(test_relation_triplets).cuda()
        test_init_emb_ent = torch.load(f"ckpt/best/{data_name}/best.ckpt")["inf_emb_ent"]
        test_init_emb_rel = torch.load(f"ckpt/best/{data_name}/best.ckpt")["inf_emb_rel"]
        emb_ent, emb_rel = my_model(test_init_emb_ent, test_init_emb_rel, msg, relation_triplets)

        head_ranks = []
        tail_ranks = []
        ranks = []
        for triplet in tqdm(sup):
            triplet = triplet.unsqueeze(dim = 0)

            head_corrupt = triplet.repeat(test.num_ent, 1)
            head_corrupt[:,0] = torch.arange(end = test.num_ent)

            head_scores = my_model.score(emb_ent, emb_rel, head_corrupt)
            head_filters = test.filter_dict[('_', int(triplet[0,1].item()), int(triplet[0,2].item()))]
            head_rank = get_rank(triplet, head_scores, head_filters, target = 0)

            tail_corrupt = triplet.repeat(test.num_ent, 1)
            tail_corrupt[:,2] = torch.arange(end = test.num_ent)

            tail_scores = my_model.score(emb_ent, emb_rel, tail_corrupt)
            tail_filters = test.filter_dict[(int(triplet[0,0].item()), int(triplet[0,1].item()), '_')]
            tail_rank = get_rank(triplet, tail_scores, tail_filters, target = 2)

            ranks.append(head_rank)
            head_ranks.append(head_rank)
            ranks.append(tail_rank)
            tail_ranks.append(tail_rank)


        print("\n========LP========")
        mr, mrr, hit10, hit3, hit1 = get_metrics(ranks)
        print(f"MR: {mr:.1f}")
        print(f"MRR: {mrr:.3f}")
        print(f"Hits@10: {hit10:.3f}")
        print(f"Hits@1: {hit1:.3f}")
```

```
100%|██████████| 2329/2329 [00:02<00:00, 790.70it/s]
========LP========
MR: 171.5
MRR: 0.223
Hits@10: 0.371
Hits@1: 0.146
```