# Artificial Intelligence - Assignment - 4

Knowledge Graph Embedding

Consider the colab notebooks:

- KGE with PyKEEN
  https://colab.research.google.com/drive/1_whF7q_LvHFM91w9MofF68dm41PcZclF?usp=share_link
- Rescal Implementation
  https://colab.research.google.com/drive/1IJkamHKZD0O0uWObQrpSrrGQrTZo-J5f?usp=share_link

# Part 1

**Compare** the Rescal implementation from PyKEEN (https://pykeen.readthedocs.io/en/stable/api/pykeen.models.RESCAL.html) with the implementation provided in the second colab notebook. Train both models using train and validation set from https://github.com/ZhenfengLei/KGDatasets/tree/master/Countries/Countries_S1 considering both the relations “locatedin” and “neighbor” (as seen during the hands-on session).

Then, considering only the relation **locatedin,** for each triple in the test set (excluding the ones with the relation “neighbor”) predict the tail of the triple. Evaluate the two models calculating:

1. **Hits@10**: it measures the fraction of predictions where the “true” tail entity is in the top 10 entities with the highest score.
2. **Mean Rank**: it measures the average rank of predictions; if the correct entity’s score is the third highest, then the rank is 3 (MR is highly dependent on the number of total entities).
3. **Mean Reciprocal Rank**: it measures the average of the reciprocal ranks (it is bounded between 0 and 1)

## An example to better clarify the three metrics

Given the following triples:

1. italy - locatedin - southern_europe
2. japan - locatedin - asia
3. us - locatedin - northern_america

We remove the tails and we ask a model to predict them.

1. italy - locatedin - X
2. japan - locatedin - X
3. us - locatedin - X

For each triple we consider the 3 entities with the highest score.
Predictions:

1. italy - locatedin - [ africa, southern_europe, western_europe, …]
2. japan - locatedin - [ asia, southern_america, indonesia, … ]
3. us - locatedin - [ asia, southern_america, southern_europe, …]

Now we calculate the Hits@3 that is the fraction of predictions where the “true” tail entity is in the top 3 ranks: this happens for prediction 1. and 2. but not for prediction 3. ⇒ Hits@3 = ⅔.

The rank of the predictions 1. and 2. are respectively 2, 1 because “southern_europe” is the second ranked entity for italy-locatedin and “asia” is the first ranked entity for japan-locatedin. For prediction 3., in this case, the rank is higher than 3, let’s suppose it is 14. Then the Mean Rank can be obtained with (2 + 1 + 14) / 3 = 5.666667.

Finally the Mean Reciprocal Rank with (½ + 1/1 + 1/14) / 3 = 0.5238095.

# Part 2

Try to find and visualize (e.g. in a table or dataframe) some example predictions that show that one implementation gives better results than the other.

# Hints

- Use the function *get_tail_prediction_df* (https://pykeen.readthedocs.io/en/stable/api/pykeen.models.predict.get_tail_prediction_df.html#pykeen.models.predict.get_tail_prediction_df ) for the PyKEEN implementation.
- To filter the dataset so that it contains only triples with the “locatedin” relation you could use: `countries_train_pd.query('relation == "locatedin"')`