algorithm/KGCN/model.py

```
from typing import List

import tensorflow as tf

from Recommender_System.algorithm.KGCN.layer import SumAggregator, ConcatAggregator, NeighborAggregator

from Recommender_System.utility.decorator import logger



@logger('初始化KGCN模型：', ('n_user', 'n_entity', 'n_relation', 'neighbor_size', 'iter_size', 'dim', 'l2', 'aggregator'))

def KGCN_model(n_user: int, n_entity: int, n_relation: int, adj_entity: List[List[int]], adj_relation: List[List[int]],

               neighbor_size: int, iter_size=2, dim=16, l2=1e-7, aggregator='sum') -> tf.keras.Model:

    assert neighbor_size == len(adj_entity[0]) == len(adj_relation[0])

    l2 = tf.keras.regularizers.l2(l2)



    user_id = tf.keras.Input(shape=(), name='user_id', dtype=tf.int32)

    item_id = tf.keras.Input(shape=(), name='item_id', dtype=tf.int32)



    user_embedding = tf.keras.layers.Embedding(n_user, dim, embeddings_initializer='glorot_uniform', embeddings_regularizer=l2)

    entity_embedding = tf.keras.layers.Embedding(n_entity, dim, embeddings_initializer='glorot_uniform', embeddings_regularizer=l2)

    relation_embedding = tf.keras.layers.Embedding(n_relation, dim, embeddings_initializer='glorot_uniform', embeddings_regularizer=l2)



    u = user_embedding(user_id)



    flatten = tf.keras.layers.Flatten()

    entities = [tf.expand_dims(item_id, axis=1)]  # [(batch, 1), (batch, n_neighbor), (batch, n_neighbor^2), ..., (batch, n_neighbor^n_iter)]
```

```
    relations = []  # [(batch, n_neighbor), (batch, n_neighbor^2), ..., (batch, n_neighbor^n_iter)]
    for _ in range(iter_size):
        neighbor_entities = flatten(tf.gather(adj_entity, entities[-1]))
        neighbor_relations = flatten(tf.gather(adj_relation, entities[-1]))
        entities.append(neighbor_entities)
        relations.append(neighbor_relations)

    if aggregator == 'sum':
        aggregator_class = SumAggregator
    elif aggregator == 'concat':
        aggregator_class = ConcatAggregator
    elif aggregator == 'neighbor':
        aggregator_class = NeighborAggregator
    else:
        raise Exception("Unknown aggregator: " + aggregator)

    entity_vectors = [entity_embedding(entity) for entity in entities]  # [(batch, 1, dim), (batch,
n_neighbor, dim), (batch, n_neighbor^2, dim), ..., (batch, n_neighbor^n_iter, dim)]
    relation_vectors = [relation_embedding(relation) for relation in relations]  # [(batch, n_neighbor, dim),
(batch, n_neighbor^2, dim), ..., (batch, n_neighbor^n_iter, dim)]
    for it in range(iter_size):
        aggregator = aggregator_class(activation='relu' if it < iter_size - 1 else 'tanh', kernel_regularizer=l2)
        entities_next = []
        for hop in range(iter_size - it):
            inputs = (entity_vectors[hop], entity_vectors[hop + 1], relation_vectors[hop], u)
            vector = aggregator(inputs, neighbor_size=neighbor_size)
            entities_next.append(vector)
        entity_vectors = entities_next
    assert len(entity_vectors) == 1
```

```
    i = tf.reshape(entity_vectors[0], shape=(-1, dim))  # batch, dim
    score = tf.sigmoid(tf.reduce_sum(u * i, axis=1))

    return tf.keras.Model(inputs=[user_id, item_id], outputs=score)


if __name__ == '__main__':
    adj = [[1, 2], [0, 2], [0, 1]]
    model = KGCN_model(3, 3, 3, adj, adj)
```