1. get\_score( : tuple, : tuple):
2. --------------------------------
3. score = 0
4. // we count both directions, for example for mapping earth🡪electrons, sun🡪nucleus
5. // we will count (earth:sun,electrons:nucleus) and (sun:earth,nucleus:electrons)
6. for each direction:
7. = // List[str]
8. = // List[str]
9. // this will create a full bipartite graph between and
10. = // List[Tuple[str, str, float]]
12. // clustering is using AgglomerativeClustering of sklearn.cluster
13. // <https://scikit-learn.org/stable/modules/generated/sklearn.cluster.AgglomerativeClustering.html>
14. // 🡪 how close the props in the cluster
15. = // Dict[int, List[str]]
16. = // Dict[int, List[str]]
17. // between every two clusters (from the opposite side of the bipartite) we will take
18. // only one edge, which will be the one with the maximum weight.
19. clusters\_edges = get\_clusters\_edges( , , )
20. // we want the maximum-weight of full bipartite matching
21. // we will use networkx algorithm of minimum\_weight\_full\_matching
22. // https://networkx.org/documentation/stable/reference/algorithms/generated/networkx.algorithms.bipartite.matching.minimum\_weight\_full\_matching.html
23. best\_matching = maximum\_weight\_full\_matching( clusters\_edges )
24. score += sum([edge[2] for edge in best\_matching])
25. --------------------------------
26. return score
27. get\_edge\_props( : string, : string ):
28. --------------------------------
29. = // sorted by plausibility
30. = // why do, why does, how do, how does
31. = // sorted by concept-net weights
32. --------------------------------
33. return
34. get\_edges\_weights( : List[string], : List[string] ):
35. --------------------------------
36. edges = []
37. for in :
38. for in :
39. // similarity is calculated by cosine-similarity.
40. // https://pytorch.org/docs/stable/generated/torch.nn.CosineSimilarity.html
42. --------------------------------
43. return