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