物理学报 Acta Physica Sinica
Chinese Physical Society
Institute of Physics, CAS

知识图谱复杂网络特性的实证研究与分析

丁连红 孙斌 时鹏

Empirical study of knowledge network based on complex network theory

Ding Lian-Hong Sun Bin Shi Peng

引用信息 Citation: Acta Physica Sinica, 68, 128902 (2019) DOI: 10.7498/aps.68.20190106
在线阅读 View online: https://doi.org/10.7498/aps.68.20190106
当期内容 View table of contents: http://wulixb.iphy.ac.cn

知识图谱复杂网络特性的实证研究与分析*

丁连红[1)] 孙斌[1)] 时鹏[2)†]

1) (北京物资学院信息学院, 北京 101149)

2) (北京科技大学国家材料服役安全科学中心, 北京 100083)

(2019 年 1 月 20 日收到; 2019 年 4 月 21 日收到修改稿)

知识图谱是人工智能研究的新热点, 用于解决智能检索、自动回答等应用问题. 概念图谱是微软发布的大型知识图谱, 本文以概念图谱为研究对象构建了概念图谱网络, 并对其特性进行了分析. 为解决概念图谱海量数据带来的计算困难, 提出一种适合概念图谱的最大子网构建方法和一种网络近似平均路径的计算方法; 对不同度值给出了不同的聚类系数求解方法, 并通过 Map/Reduce 模式进行提速. 结果表明: 概念图谱网络具有无标度和极端小世界网络的特征; 平均路径长度随网络规模增加而减小并趋于 4 这一定值, 网络的"菱形"结构能很好地解释这一现象; 概念图谱网络是异构的, 相邻节点的度负相关; k-shell 分解表明子概念占核心层节点的 99.5%, 在概念图谱中起重要的连接作用; 子概念的缺失对概念图谱的完整性影响最大、概念其次、实例最小; 82% 的实例节点度为 1, 40% 的概念节点度为 1, 实例比概念更不易因一词多义而引起理解歧义.

关键词: 微软概念图, 复杂网络, 子网抽取, 近似平均路径

PACS: 89.75.Fb, 05.90.+m, 02.10.Ox **DOI:** 10.7498/aps.68.20190106

1 引 言

复杂网络理论兴起于 20 世纪 90 年代, 是对复杂系统的一种抽象和描述方式. 复杂网络由节点和边组成, 节点表示元素, 边表示元素之间的互相作用. 复杂网络普遍存在于现实世界, 如生物分子网、互联网、交通网、电力网等.

知识图谱由 Google 公司于 2012 年提出后, 就在搜索引擎与人工智能领域备受关注. 知识图谱旨在通过语义知识的结构化储备实现智能检索、自动问答等应用, 相关研究主要围绕着知识图谱的知识提取、知识融合以及知识推理等进行. 知识图谱描述知识之间的语义关联, 同样可以网络化. 网络化的知识图谱是否符合复杂网络模型? 知识间的关系是否可以通过复杂网络理论进行分析? 目前尚没有明确的结论. 微软概念图谱 (Microsoft concept graph) 是微软研究院于 2016 年发布的一个知识图谱, 其形式为概念 (concept)、实例 (instance) 和关系 (relation) 的三元组, 描述实例与概念之间的 IsA 关系, 用于实现实例的概念化推理 [1]. 例如三元组 (sport, basketball, 6423) 表示 basketball 与 sport 之间存在 IsA 关系, 即 basketball 是一种 sport. 其中 basketball (篮球) 是实例, sport (运动) 是概念, 6423 是从微软 bing 搜索日志中抽取到的 basketball 和 sport 之间的 IsA 关系的次数, 以下称为共现次数. 概念和实例是对事物的描述, 概念较抽象, 通常描述事物的类别, 例如

* 国家重点研发计划(批准号: 2017YFB0203703)、国家自然科学基金重点项目(批准号: 71831001)、北京市教委科技计划一般项目(批准号: KM201910037186)、北京市教委社科计划一般项目(批准号: SM201610037001)和北京物资学院高水平科研团队协同基金(批准号: 2017GG05)资助的课题.

† 通信作者. E-mail: shipengustb@sina.com

fruit (水果); 实例描述较为具体, 一般为具体事物的名称或标识, 例如 orange (橙子). 概念图谱正是通过这些由用户搜索记录提取到的实例、概念以及它们之间的 IsA 关系反映人们对实物的认知和理解.

本文以微软概念图谱为研究对象, 构建描述概念与实例之间概念化关系的概念图谱网络, 进而研究概念词与实例词之间的关联关系. 选取该研究对象的优势在于: 1) 图谱数据来源于 bing 搜索日志中数以亿计的用户搜索记录, 反映了用户对事物的真实看法; 2) 图谱中的共现次数反映了 IsA 关系的可信度, 可据此调整网络规模; 3) 微软概念图谱只有 IsA 一种关系, 在构造网络时不需要对关系进行区分, 简化了网络模型的构建.

首先采用 NetworkX 工具计算相关统计特征[2], 但由于概念图谱网络的节点数量巨大, 导致计算时间过长, 因此本文提出一种适用于概念图谱的子网提取算法, 用来获取最大连通子网, 在时间和空间效率上都有显著提高, 并对最大子网的复杂网络特征进行了分析. 在计算网络的最短平均路径长度时, 本文提出了一种计算概念图谱网络近似平均最短路径的方法, 并对算法准确性进行了验证. 对于聚类系数的计算, 本文根据节点度值给出了不同的聚类系数求解方法, 并采用 Map/Reduce 模式进行了计算提速.

2 相关工作

复杂网络理论起步于 Erdos 和 Renyi 的随机图论 (ER 图), 而后随着 Watts 和 Strogatz[3] 提出的小世界网络、Barabási 和 Albert[4] 提出的无标度网络逐渐兴起, 复杂网络理论为复杂系统的特征分析提供了重要的理论依据, 已被广泛应用于社会网络分析[5]、城市建设用地布局[6]、国际贸易交流分析[7]、交通运输分析[8] 等. 其中交通运输领域的应用又包括公路交通[9]、轨道交通[10]、铁路[11]、航运[12]、航空[13] 等. 通过求解网络特征, 如网络节点的度分布、平均路径长度、聚集特性以及介数中心性等, 分析复杂系统中各因子之间的关系和整体稳定性[14]. 以上文献中复杂网络系统的节点以交通站点、城市、国家等为主, 边以交通线路、城市区域间道路、国家关系等为主, 系统中节点和边的数量较少, 且多为连通网络, 因此分析其统计特性时资源消耗不大.

在研究如万维网[15,16]、电话呼叫网[17,18] 这类超大规模网络时, 由于很难得到描述整个网络结构的全部信息, 通常的方法是先分析局部实际数据的特性, 根据这些特性建立实际网络的数学模型, 再由数学模型推算整个网络的相关特性. Albert 等[16] 先从万维网抓取了部分实际数据, 得到其局部结构, 据此分析出节点的度分布符合幂律分布, 并根据拟合结果对局部网络进行扩展得到较大规模的拓扑结构; 据此分析万维网的拓扑结构和连接特性, 得到网络半径与节点数量之间的函数关系; 最后, 将万维网节点实际数量代入该函数推测出万维网的网络直径. Aiello 等[17,18] 也通过类似的方法对电话呼叫网进行了拓扑建模和演化研究. 目前较快的串行最短路径算法的时间复杂度也只能降低到 $O(n^{2.376})$[19], n 为网络节点的数量.

随着时间的发展, 万维网的节点数量早已超过数十亿数量级, 边的数量更是不计其数; 电话呼叫网络根据已存在的电话线路之间的关系进行建立, 包括约 47000000 个节点, 80000000 条边. 对于规模如此巨大的网络直接计算其直径 (节点间最短路径的平均值) 和聚类系数几乎是不可行的. 这可能是相关文献均未给出具体的网络聚类系数值, 对于网络直径也只给出由网络模型计算得到的推测值的原因[15,17,18].

因此有学者开展了网络精简、评估方法改进、网络结构优化等研究. 网络精简通过阈值网络[20]、最小生成树[21]、差分网络[22] 等方法减小网络的规模. 孙延风和王朝勇[23] 采用互信息表示金融网络中节点间的关系, 并用阈值网络和最小生成树对网络进行精简, 最后验证了网络模型的有效性.

评估方法改进包括基于度中心性的评估方法[24]、k-shell 分解[25] 和基于 PageRank 的评估方法[26] 等. Ruan 等[27] 将约束系数引入节点核心性的度量, 提出了一种综合节点邻居节点的 k-shell 值和邻居节点间拓扑结构的核心性度量方法, 该方法能更准确地评估节点的传播能力. 孔江涛等[28] 利用复杂网络动力学模型描述复杂网络中节点间的相互影响活动, 建立了基于偏离均值与偏离均值的方差的两级节点重要性评估标准, 并通过扰动测试和破坏测试验证了标准的有效性.

网络结构优化主要以实际应用为目的, 如韩定定等[29] 结合时变特征和网络客流分布, 对航空网进行优化, 提出了一种快速推算航线网络最优拓扑

及相应航班频率分布的方法. Niu 和 Pan[30] 通过自组织优化机制对运输网络进行优化, 证明了无标度网络在 gradient-driven 运输模式下可以有效地通过自组织优化机制提高运输能力. Jiang 等[31] 以中国航空运输网 (CNTA) 为例研究多层网络融合过程的本质, 通过一系列拓扑属性刻画多层网络的融合过程, 发现 CNTA 在融合过程中发生了明显的结构转换, 为中国航空运输系统的网络结构优化和管理提供了启示.

而知识图谱作为智能化的语义知识储备, 其规模也会随着时间而不断积累扩大, 节点数量往往突破千万数量级, 对这类大规模网络的特征计算也必然十分复杂与耗时. NetworkX 是一款用 Python 语言开发的复杂网络构建与分析软件工具包, 是复杂网络建模与分析的有力工具[2]. 由于知识图谱的规模巨大, 使用 NetworkX 仅是构建出整个网络便需要消耗近 40 GB 的内存. 万维网可以通过网络中的路由设备找到通往各个网站节点需要跳转的次数, 即平均最短路径长度, 而知识图谱与万维网不同, 知识间的拓扑距离无法如此计算. 为了解决大规模复杂网络分析的问题, 有学者设计了近似算法以计算复杂网络的平均距离. Rattigan 等[19] 通过引入网络结构索引 (network structure index, NSI) 计算节点间路径, 从而降低计算网络平均路径长度这样的基于节点间路径应用的搜索复杂度. NSI 由各个节点的标记和根据节点标记估算节点对间距离的函数构成. 唐晋韬等[32] 提出了基于区域中心点距离 (centers distance of zone, CDZ) 的复杂网络最短路径计算的近似方法, 根据网络特征将网络分成多个区域, 每个区域都设置一个中心点, 将两节点之间的距离转换为节点到中心点以及中心点间的距离之和, 实现近似计算. 受 CDZ 思想的启发本文提出了一种根据节点所在网络层与中心节点的距离及不同网络层之间的距离计算概念图谱网络近似平均最短路径的方法.

3 概念图谱网络构建及最大连通子网抽取

3.1 概念图谱网络构建

微软官方提供的概念图谱的数据文件是一个纯文本文件, 构成该文件的元数据是描述实例与概念之间 IsA 关系的三元组. 通过对该数据文件的遍历发现, 有些词既作为概念 (Concept) 出现, 也作为实例 (Instance) 出现, 本文将这部分词称为子概念词 (subConcept). 为了构建描述实例与概念之间的实例关系的复杂网络, 将概念、子概念和实例作为网络的节点, 在存在 IsA 关系的两个节点之间添加一无向条边, 表示它们之间的实例关联, 从而构建出如图 1 所示的描述概念、子概念、实例之间的实例关联的概念图谱网络.

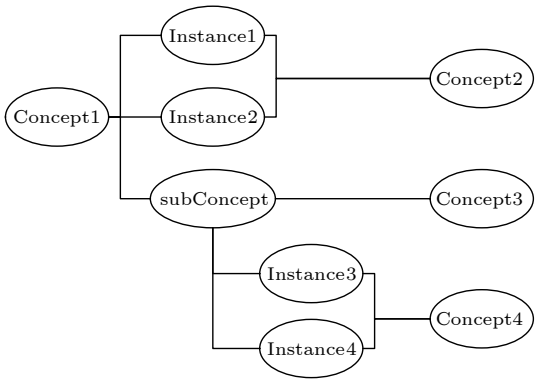

图 1 概念图谱网络示意图

Fig. 1. Network of concept graph.

建立的概念图谱网络是一种无向非加权网, 具有如下特征: 1) 网络规模巨大, 包括 16936670 个节点和 33354328 条边, 因此建立非加权网络以减少计算复杂度, 便于特征分析; 2) 只描述概念和实例间的 IsA 关联及关联间的跳转层次, 忽略关系的方向; 3) 共现次数描述节点间关系的可信度, 旨在通过阈值设置得到不同较小规模的概念图谱网络以进行深入的比较分析.

表 1 列出了在整体概念图谱网络中, 概念词、实例词以及子概念词的度值分布. 在概念图谱网络中, 概念节点的度表示有多少个实例或子概念与该概念存在 IsA 关系. 实例节点的度表示该实例与多少个概念或子概念具有 IsA 关系. 首先, 因为概念图谱的基本数据是由概念、实例和它们之间的 IsA 关联构成的三元组, 因此, 每个概念至少与一个实例相关, 一个实例也至少与一个概念相关, 因此所构建的概念图谱网络中不应有孤立节点, 即不存在度为 0 的节点, 这与表 1 的统计数据一致. 其次, 由于子概念既可以处在概念的位置, 也可以处在实例的位置, 所以子概念的度都应大于等于 2. 但由表 1 可知概念图谱网络中有 18 个子概念节点的度为 1. 为探究该现象, 在概念图谱的数据文件

中对度为 1 的子概念节点进行了检索和分析, 发现存在类似于 (small business issuer, company, 4) 和 (company, small business issuer, 1) 这样的两个节点互为对方的概念与实例的情况. 同时, 因为这两个节点都未出现在其他三元组中, 所以它们的度都为 1. 这些度为 1 的子概念节点也揭示了所构建的概念图谱网络并非一个连通网络. 因此, 为了真实描述其网络特性, 必须抽取该网络的最大连通子网, 并将其作为后续研究对象分析概念图谱网络的复杂网络特性.

表 1 概念图谱网络的节点度分布

Table 1. Degree distribution of the concept graph network.

k	Concept		k	Instance		k	SubConcept	
	Quantity	Proportion		Quantity	Proportion		Quantity	Proportion
1	2061496	0.464809	1	9610146	0.831317	1	18	1.91208×10^{-5}
2	1165725	0.262838	2	1014355	0.087746	2	140735	0.149498132
3	538652	0.121451	3	312310	0.027016	3	186330	0.197932191
4	252438	0.056918	4	156703	0.013555	4	125273	0.133073361
5	130975	0.029531	5	96100	0.008313	5	78365	0.083244546
6	74760	0.016856	6	64809	0.005606	6	52113	0.055357915
7	46336	0.010447	7	46409	0.004015	7	37615	0.039957169
8	31509	0.007104	8	34801	0.00301	8	29314	0.031139292
9	22506	0.005074	9	27177	0.002351	9	23419	0.024877229
10	16510	0.003723	10	21928	0.001897	10	19484	0.020697208
11	12921	0.002913	11	18095	0.001565	11	16465	0.017490224
12	10012	0.002257	12	14935	0.001292	12	14188	0.015071443
13	8188	0.001846	13	12688	0.001098	13	12491	0.013268776
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
32773	1	2.25×10^{-7}	6716	1	8.65×10^{-8}	364276	1	1.06227×10^{-6}
Total	4435143	1	Total	11560144	1	Total	941383	1

3.2 子网抽取算法

由于概念图谱网络不是连通网络, 无法计算节点间的平均距离, 所以需要计算出概念图谱网络的最大连通子网. 我们按照广度优先思想, 实现了基于广度优先的子网提取算法 (SubNet Extraction based on Breadth-first, SNEBF).

算法 1 基于广度优先的子网提取算法 (SNEBF)

输入: 概念图谱数据文件 Graph

输出: 最大连通子网节点集合 MaxSubNet

1) 从 Graph 中抽取一个三元组, 该三元组包含的两个节点的度之和最大, 将该三元组中的概念和实例作为两个初始节点, 构成初始节点集合 InitialSet, 并将这两个节点加入 MaxSubNet.

2) 对 InitialSet 中的每一个节点 node

{a. 遍历 Graph 中的三元组 (概念 (con), 实例 (ins), 关系 (rel)) 信息, 如果其 con 或 ins 等于 node, 则将该三元组中相应的 ins 或 con 加入节点 node 的邻居节点集合 NeighborSet.

b. 判断 NeighborSet 中是否还有未加入 MaxSubNet 的节点, 如果有, 将这些节点加入 MaxSubNet, 将 NeighborSet 集合与 MaxSubNet 的差集并入 InitialSet.}.

3) 返回 MaxSubNet.

类似于广度优先的按层扫描, 对于 InitialSet 中的每一节点 node 算法 1 总是找出其全部相邻节点并据此对最大子网进行扩展, 对找出的相邻节点重复相同的操作. 但在实际运行时, 由于网络中的节点之间关系复杂, 不同节点的邻居节点集合可能相互包含了许多相同的节点. 如在图 2 所示的概念图谱网络中, Concept1 的邻居节点集合包括 4 个节点: Instance2, Instance3, Instance4 和 Instance5. Concept2 也有 4 个邻居节点: Instance1, Instance2, Instance3 和 Instance4. 其中 Instance2, Instance3

和Instance4 是Concept1 的邻居节点也是Concept2 的邻居节点.

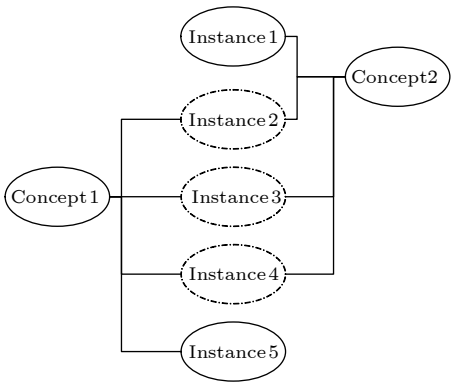

图 2 导致节点的邻居节点集合冗余的网络结构

Fig. 2. Network structure leading to the overlap of neighbor node sets.

这些相同的节点在处理 Concept1 的邻居节点和 Concept2 的邻居节点时会被重复进行递归检索处理. 这不但会增加大量的冗余存储空间而且需要多次重复检索图谱数据. 同时由于算法中循环嵌套了循环, 所以在进行遍历时十分耗时. 因此我们引入集合运算, 从而得到基于集合运算的子网抽取算法 (SubNet Extraction based on Set Operation, SNESO).

算法 2 基于集合运算的子网抽取算法 (SNESO)

输入: 概念图谱数据文件 Graph

输出: 最大连通子网节点集合 MaxSubNet

1) 从 Graph 中抽取一条包含度值最大节点的三元组信息, 将该三元组中的概念和实例作为中心节点构成最大连通子网的中心层 SubNet_i (此时 $i=1$), 并将其加入 MaxSubNet.

2) 寻找 SubNet_i 集合的邻居节点, 即遍历 Graph 中的 (con, ins, rel) 信息, 判断其 con 或 ins 是否属于 SubNet_i 集合, 若存在, 则表示对应的 ins 或 con 是集合 SubNet_i 的邻居节点, 将这些节点加入 SubNet_i 的邻居节点集合 $\mathrm{NeighborsSet}_i$.

3) 判断 $\mathrm{NeighborsSet}_i$ 中是否还有新的未在 MaxSubNet 中的节点, 如果有, 则将 $\mathrm{NeighborsSet}_i$ 并入 MaxSubNet, 将 SubNet_i 替换为 $\mathrm{NeighborsSet}_i$ 与 MaxSubNet 的差集, $i=i+1$. 然后跳转至步骤 2; 若无, 则继续步骤 4.

4) 返回 MaxSubNet.

算法 1 和算法 2 的关键操作是对 Graph 三元组信息的遍历. 由中心层出发, 每扫描一次 Graph, 算法 1 获得一个节点的邻接节点, 算法 2 获得一层节点的全部邻接节点, 因此算法 2 能显著减少对 Graph 的扫描次数. 由于度值大的节点在最大连通子网的概率较高, 为了保证可以找到最大子网, 我们从度值最大的节点以及与最大节点相连的度值较大的节点开始进行搜索. 在实际运行中, 运算时间和空间复杂度确实有所降低, 具体的复杂度分析见 3.3 节.

先由算法 2 得到最大连通子网的节点集合 MaxSubNet; 再根据 MaxSubNet 从概念图谱数据文件 Graph 中提取出最大子网的边的集合, 其中的每条边依然采用 Graph 中的三元组形式并存储在文本文件 GraphLink 中; 最后由 MaxSubNet 和 GraphLink 生成如图 3 所示结构的最大连通子网. 其中, 中心层 (Central Layer) 由两个节点构成, 这两个节点是各三元组对应的节点对中度值和最大的节点对; 第二层 (Layer-2) 为中心层各节点的邻居节点的集合; 第三层 (Layer-3) 为第二层的各节点的邻居节点的集合. 以此类推, 直到网络的最外层 (Outermost Layer).

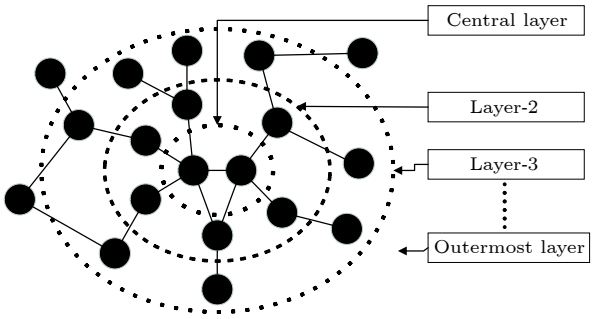

图 3 最大连通子网的分层逻辑结构

Fig. 3. Hierarchical logical structure of the largest connected subnet.

3.3 算法复杂度分析

根据算法 1 我们实现了函数 GetNeighbor (Node, Graph), 该函数遍历 Graph 中的边 (三元组) 信息, 判断 Node 是否为当前边的端点. 由于一条边有两个端点, 完成一条边的扫描需要两次判断操作. 将 1 个端点的判断操作定义为 1 次基本操作, Graph 的边数为 n, 则 GetNeighbor(Node, Graph) 的时间复杂度为 $2n$. 由算法 1 可知, 每当

最大连通子网中有一个新的节点, 就要调用一次 GetNeighbor(Node, Graph). 设最大连通子网的节点数为 m, 则算法 1 的时间复杂度为 $m \times 2n$. 由后面 3.4 节的实际计算结果可知, m 近似为 $n/2$, 所以算法 1 的时间复杂度为 $O(n^2)$.

算法 2 在算法 1 的基础上引入了集合运算, 两者的区别在于: 算法 1 每遍历一次 Graph 可以找出一个节点的邻居节点; 算法 2 每遍历一次 Graph 可以找出一层节点的全部邻居节点, 只要将函数 GetNeighbor 的参数由 Node 改为 NodeSet 即可. 该函数遍历 Graph 中的边信息, 判断边端点是否在集合 NodeSet 中. 由于 GetNeighbor(Node, Graph) 的时间复杂度为 $2n$, 由 3.4 节的实验可知 GetNeighbor(NodeSet, Graph) 的平均运算时间约为算法 1 中 GetNeighbor(Node, Graph) 运行时间的 1.6 倍, 所以 GetNeighbor(NodeSet, Graph) 的时间复杂度为 $1.6 \times 2n = 3.2n$. 由算法 2 可知, 对最大子网的每一子网层需调用一次 GetNeighbor (NodeSet, Graph), 设构建最大连通子网过程中, 子网层数为 nl, 则运行 GetNeighbor(NodeSet, Graph) 的次数为 nl 次, 所以算法 2 的时间复杂度为 $nl \times 3.2n$. 通过 3.4 节的最大子网抽取实验发现, nl 的实际值为 12, 由于 nl 远小于 n(33377320), 所以算法 2 的时间复杂度近似为 $O(n)$.

3.4 最大子网抽取实验与对比

分别采用 NetwrokX 和算法 1(SNEBF)、算法 2(SNESO) 对所构建的概念图谱网络进行了最大连通子网的抽取实验, 具体的实验环境为 64 位 Win7 系统, 16 GB 内存, Anacodna 集成 Python 开发环境, 其时间复杂度和实际运算时间见表 2.

表 2 最大子网提取算法时间复杂度对比表

Table 2. Time complexity of the subset extraction algorithms.

Algorithm	Parameters		Time complexity	Time cost
NetworkX	—	—	—	15 d以上
SNEBF	$m = 15\ 114\ 834$	$n = 33\ 377\ 320$	$m \times 2n = 15\ 114\ 834 \times 2n$	约5.22 a
SNESO	$nl = 12$	$n = 33\ 377\ 320$	$nl \times 3.2n = 19.2 \times 2n$	3.49 min (实际运算3.80 min)

其中采用 NetworkX 以及 SNEBF 求解最大子网时程序并未运行出结果, 在实际运行时间为 15 d 时终止了这两个程序, 由此可以得出 NetworkX 构建最大子网时间大于 15 d 的结论. 实验过程中 SNEBF 中的 GetNeighbor (Node, Graph) 运算时间约为 10.9 s, SNESO 中的 GetNeighbor (NodeSet, Graph) 的平均运算时间为 17.6 s, 约为 GetNeighbor (Node, Graph) 的 1.6 倍. 因为 SNEBF 的时间复杂度为 $(m \times 2n)$, 而 $2n$, 即 GetNeighbor(Node, Graph) 的平均运行时间为 10.9 s, 所以可以推算出 SNEBF 的运行时间约为 $m \times 2n = 15114834 \times 10.9\ \text{s} = 5.22$ a. 由 GetNeighbor(NodeSet, Graph) 的平均计算时间和 $nl = 12$, 得到 SNESO 的运行时间为 3.49 min, 而该算法的实际运行时间为 3.80 min. 所以, SNESO 时间复杂度最小, 计算速度最快, 运算时间在可接受范围内, 远小于 SNEBF 的运行时间.

使用 NetworkX 抽取最大子网, 首先通过其内部函数 connected_component_subgraphs(Graph) 求解所有子网 [33], 再从中找到节点数量最多的子网, 即为最大连通子网. connected_component_subgraphs(Graph) 的输入是一个无向 NetworkX 图 G, 输出是 G 的所有连通子网的列表. 该函数是一个由两行代码构成的 for 循环:

for c in connected_components(G):

yield G.subgraph(c).copy()

其中 connected_components(G) 返回各个连通子网的全部节点 c 的列表, G.subgraph(c).copy() 将节点列表 c 转换成图 G 的一个连通子图. 以下是 connected_components(G) 的定义:

① seen={};

② for v in G: /*对 G 中的每个节点 v*/

③ if v not in seen: /*如果没有遍历过节点 v*/

④ c = sp_length(G, v); /*计算点 v 到所有可以连通节点的距离字典 c*/

⑤ yield list(c); /*将获得的 c 转换为列表返回*/

表 3 算法空间复杂度和实际内存消耗对比表

Table 3. Space complexity of the algorithms.

Algorithm	Parameters	Space complexity	Memory cost
NetworkX	—	—	40 GB
ESNSO	SubNet_i, NeighborsSet, MaxSubNet	31724479	5.23 GB

⑥ seen.update(c);/*更新 seen 的值以确保不会寻找重复节点*/

⑦ end if

⑧ end for;

NetworkX 求解最大子网的关键操作是 sp_length(G, v) 函数, 具体的时间复杂度分析在第 4 节中采用事后统计法给出.

表 3 列出了 NetworkX 与算法 2 运行时的实际内存消耗. 实验中 NetworkX 抽取概念图谱最大连通子网至少需要 40 GB 内存. 采用 SNESO 求解最大子网时, 占用内存的变量为 SubNet_i, NeighborsSet, MaxSubNet, 数值与最大子网的节点数、子网某层的节点数相关, 实际占用内存为 5.23 GB.

由算法 2 根据概念图谱的数据文件生成的最大子网包含 15114834 个节点和 32274081 条边, 分别占整个概念图谱网络节点和边的 89.24% 和 96.69%. 可知该子网覆盖了概念图谱网络绝大部分的节点与边, 其网络特性可以较好地反映出整个网络的特征, 后续对概念图谱复杂网络特性的分析都基于此最大连通子网.

4 概念图谱网络的复杂网络特性分析

复杂网络的统计特征主要包括度分布、k-shell 核心性、平均路径、聚类系数和度相关性等.

1) 度分布 $p(k)$: 度分布可以用来表征网络最基本的拓扑特性. 图 4 是最大连通子网的节点累积度分布, 节点度呈幂律分布, 符合无标度网络特性.

表 4 统计了三类节点中度值为 1—13 的节点数量与比例: 82% 的实例节点度值为 1, 即 82% 的实例只与一个概念相关. 度值为 1 的节点, 实例占 85.5%, 概念占 14.5%. 由此判断在自然语言处理过程中使用实例词比使用概念词更不易因一词多义而引起文本描述的歧义. 82% 的概念度值在 1—3 之间, 表明绝多数概念只有 1—3 个含义. 在度值相同的节点中, 子概念的比例随度值的增大而增加, 表明子概念通常作为连接多个节点的核心节点.

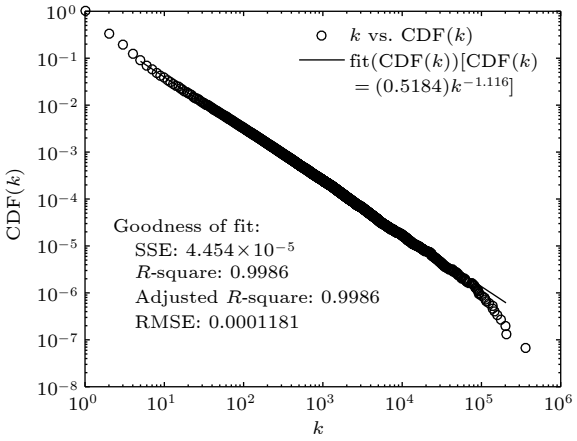

图 4 概念图谱最大连通子网累积度分布

Fig. 4. Cumulative degree distribution of the largest connected subnet.

2) k-shell 核心性: 思想是处于网络核心位置的节点, 即使度很小, 对网络的影响力也可以很大. k-shell 分解把网络由边缘至中心划分成若干层, 能够有效识别网络核心.

概念图谱最大子网经 k-shell 分解为 185 层, 图 5 是各层节点的度分布及平均度值, 可见节点度越大其 k-shell 分层越高, 越靠近中心层, 说明大度值节点位于靠近概念图谱网络中心的位置, 而不是边缘位置. 如图 5 所示, 度高于 10000 的节点大都划分到了核心层, 只有极少数 k-shell 值很低. 我们发现这些节点的多数邻居节点的度很低. 如划分到 13-shell 层的节点 "common search term" 和 "connected tool" 的度高达 102033 和 31963, 但这两个节点的邻居节点中度为 1 的分别多达 101892 个和 31942 个.

核心层 (185-shell) 包括 786 个节点, 其中子概念 782 个, 概念和实例各 2 个. 核心层节点间有 119162 条边, 构成了一个稠密图, 网络密度 0.386, 远高于最大子网的 2.825×10^{-7}. 表 5 是核心层中度值最高的 20 个节点及其度值.

虽然子概念只占最大子网节点的 6.2%, 却占核心层的 99.5%. 说明子概念在概念图谱中起着重要的连接作用. 其根源在于子概念既可以作为比其描述能力抽象的词的实例, 也可以作为比其描述具

表 4 概念图谱最大子网度分布分析

Table 4. Degree distribution analysis of the largest connected subnet.

k	Concept		Instance		SubConcept		Total
	Quantity	Percentage	Quantity	Percentage	Quantity	Percentage	Quantity
1	1468308	40.3	8636049	82.0	0	0.0	10104357
2	1014641	27.9	968156	9.2	138875	14.8	2121672
3	503109	13.8	309716	2.9	185665	19.8	998490
4	242308	6.7	156248	1.5	125045	13.3	523601
5	127833	3.5	96027	0.9	78311	8.3	302171
6	73429	2.0	64778	0.6	52090	5.6	190297
7	45834	1.3	46398	0.4	37604	4.0	129836
8	31237	0.9	34799	0.3	29301	3.1	95337
9	22401	0.6	27173	0.3	23417	2.5	72991
10	16439	0.5	21925	0.2	19488	2.1	57852
11	12880	0.4	18095	0.2	16464	1.8	47439
12	9981	0.3	14934	0.1	14187	1.5	39102
13	8169	0.2	12688	0.1	12503	1.3	33360
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
Total	3639631	···	10536663	···	938540	···	15114838

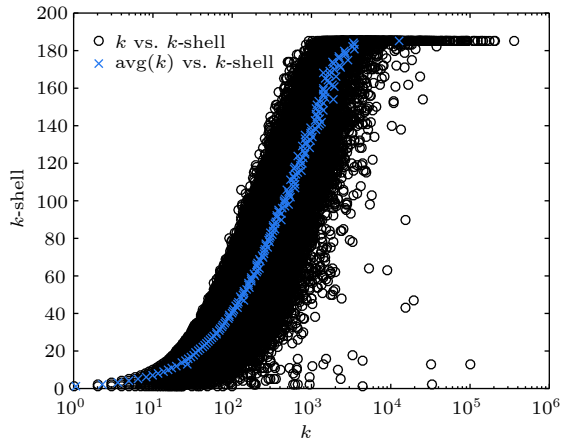

图 5 节点度与 k-shell 分解中心性关系

Fig. 5. Relationship between degree and k-shell.

表 5 核心层中度值最高的 20 个节点

Table 5. Top 20 nodes with the highest degree in core.

node	k	node	k
factor	364343	Event	113364
feature	204130	company	110609
issue	202331	program	93963
product	174283	technique	92341
item	159164	application	90644
area	144595	organization	90605
topic	137781	Name	87637
service	137398	Case	85863
activity	124670	method	84157
information	114500	project	82122

体的词的概念, 如 topic 作为概念时包括 cultural, political, physica 等实例, 这些实例都可以说是某一种具体的 topic (话题), 都与 topic 具有 IsA 关系. 作为实例时, topic 又是多个概念的实例, 如 concept, document, information 等.

3) 平均路径: 网络的平均路径 avg(l) 是所有节点对之间距离的平均值, 它描述了网络中节点间的分离程度. 目前最快的串行最短路径算法只能将计算的时间复杂度降到 $O(n^{2.376})$[19]. 概念图谱网络的节点数为 15114834, 要精确计算平均最短路径, 需要计算 114229095866361 个节点对之间的最短路径, 计算量巨大; 同时计算过程中每条路径上需要存储多个节点信息, 存储消耗也很大.

首先尝试用 NetworkX 计算最大子网的 avg(l), 运行了 30 d 没有输出结果. 为了探究其时间复杂度, 需要对较小规模的概念图谱网络进行计算. 为此将共现次数作为阈值限制边的数量从而形成不同较小规模的网络, 如表 6 所列. 其中 t 为阈值, n 为节点数, e 为边数.

图 6 展示了 NetworkX 计算表 6 中各网络平均路径的实际运算时间 time(s) 与网络规模 (节点数 n), 经过拟合可知存在函数关系: time(n) =

表 6 部分阈值网络及节点数

Table 6. Threshold networks and the number of nodes.

t	n	e	t	n	e	t	n	e	t	n	e
10	415491	936536	60	27654	66704	150	9492	19845	600	1788	2637
20	119367	303323	70	23089	54533	200	6850	13315	700	1453	2059
30	66272	169774	80	19567	45510	300	4336	7566	800	1076	1487
40	45410	114629	90	17042	38921	400	3013	4920	900	922	1222
50	34467	85085	100	15086	33983	500	2318	3577	1000	770	994

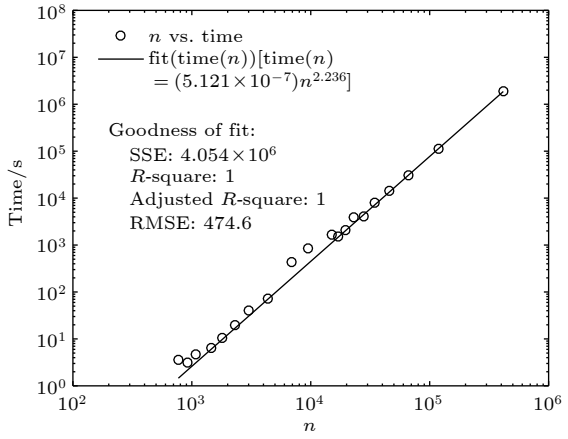

图 6 NetworkX 计算平均路径所需时间

Fig. 6. Time cost of NetworkX for avg(l).

$5.121 \times 10^{-7} \times n^{2.236}$. 当 $t = 10$ 时, NetworkX 实际运行了 1868428.416 s, 约 22 d. 将最大子网节点数代入该函数可知 NetworkX 需要计算 184 a, 这也是计算 30 d 没有输出结果的原因.

随后, 尝试用唐晋韬等[32]提出的 CDZ 方法进行计算. 该方法首先根据局部中心性寻找区域中心点并据此对网络进行区域划分; 之后用区域中心点的距离表示区域间节点的路径长度, 此时节点间路径近似等于区域中心点的路径与区域内路径的和. 其时间复杂函数为 $O(n\log n + e + d^3)$, n 是节点数, e 是边数, d 是中心点数量[32]. 相对于随机网络, CDZ 更适合具有无标度特征的复杂网络. CDZ 计算无标度网络 Cora 平均路径的时间为 20 余秒. Cora 的 $n = 30751$, $e = 134450$, 按照文献所述方法计算得到的 d 为 46, 因此时间复杂度函数值为 369792. 对于概念图谱最大子网而言, $n = 15114834$, $e = 32274081$, $d = 130109$, 因此时间复杂度函数值为 2202531075674600, 是 Cora 的 5956132434 倍. 按照 CDZ 计算 Cora 平均路径用时为 20 s 计算, CDZ 计算概念图谱的最大子网平均路径需要 3777 a.

为此我们提出了一种概念图谱网络最短路径长度的近似计算方法, 该算法区别于 CDZ 的是用网络层代替区域, 且忽略同层内节点的具体距离. 根据图 3 所示的最大连通子网的层次结构, 用层与层之间的距离近似代替点与点之间的距离, 计算网络的近似平均最短路径长度:

$$\mathrm{AppAvg}\,(l) = \frac{\mathrm{minavg}\,(l) + \mathrm{maxavg}\,(l)}{2}, \tag{1}$$

其中 AppAvg(l) 表示网络的近似平均最短路径长度, minavg(l) 表示可能存在的近似平均最短路径长度的最小值, maxavg(l) 表示可能存在的近似平均路径长度的最大值.

$$\mathrm{minavg}\,(l) = \frac{\sum_{i=1}^{n}\sum_{j=i+1}^{n}(l_{\mathrm{min}\_ij})}{n\times(n-1)/2}, \tag{2}$$

$$\mathrm{maxavg}\,(l) = \frac{\sum_{i=1}^{n}\sum_{j=i+1}^{n}(l_{\mathrm{max}\_ij})}{n\times(n-1)/2}, \tag{3}$$

其中 l_{min_ij} 表示节点 i 到节点 j 可能的最小路径长度, l_{max_ij} 表示节点 i 到节点 j 可能的最大路径长度.

由于网络的层级关系, 节点 i 到节点 j 之间必定存在一条路径为 (节点 i, 中心节点, 节点 j), 此路径为可能存在的最大的近似路径, 其距离公式为

$$\mathrm{min}\,(l_{ij}) \leqslant l = l_{i\mathrm{Center}} + l_{\mathrm{Center}j}, \tag{4}$$

其中 $l_{i\mathrm{Center}}$ 和 $l_{\mathrm{Center}j}$ 分别表示节点 i 到中心节点 (Center) 的距离和 Center 到节点 j 的路径长度. 根据对称性, 节点 i 到节点 j 与节点 j 到节点 i 的路径长度相同.

$$l_{\mathrm{Center}i} = l_{i\mathrm{Center}} = \mathrm{floor}\,(i) - 1 + 1 = \mathrm{floor}\,(i), \tag{5}$$

其中 floor(i) 表示节点 i 所在的层数. 由于中心层包括两个节点, 所以在计算时统一为节点所在的层数减去 1 再加 1. 减 1 表示节点所在层数到中心层的路径长度, 加 1 表示中心层两个节点之间的路径长度.

表 7 为经过算法 2 的运行结果后统计的网络层数及各层包含的节点数量. 根据 (4), (5) 式以及表 7 计算得到子网中各节点之间的最大距离的和为 770350922065817, 代入 (3) 式 maxavg(l) = 6.74.

表 7 子网结构与节点数量

Table 7. Subnet structure and quantity of nodes.

Layer	Quantity	Layer	Quantity	Layer	Quantity	Layer	Quantity
1	2	4	4639826	7	11921	10	73
2	553406	5	639119	8	1609	11	16
3	9202185	6	66347	9	327	12	3

假设每层节点到其他各层节点的最短距离为层数相减的绝对值, 此时, 节点对之间的路径长度最短, 有

$$l_{\min\_ij} = |\text{floor}\,(i) - \text{floor}\,(j)\,| \leqslant \min\,(l_{ij})\,. \quad (6)$$

根据 (6) 式及表 7 中的数据可以求得子网节点间最小距离的和为 125617583016439, 将其代入 (2) 式可知最小近似平均最短路径长度 minavg(l) 为 1.10. 所以子网的实际平均最短路径长度处于 (1.10, 6.74) 的开区间内, 根据 (1) 式可计算得到网络的近似平均最短路径长度为 3.92, 表明知识图谱网络中的实体平均经过 3.92 个实体就可以到达任意实体的位置, 概念图谱网络具有小世界的特性. 相对于逐条匹配而言, 以基于网状拓扑结构进行的知识搜索更为迅速.

图 7 是由 NetworkX 和本文方法计算的不同规模网络的实际平均路径 RealAvg(l) 和近似平均路径 AppAvg(l), n 为节点数量. AppAvg(l) 与 RealAvg(l) 变化趋势相同, 且随网络规模增加而减小, 并稳定在一个 4 左右的定值. 我们计算了各规模网络平均路径的真实值与近似值比值的平均值, 有 RealAvg(l) ≈ 1.1 × AppAvg(l).

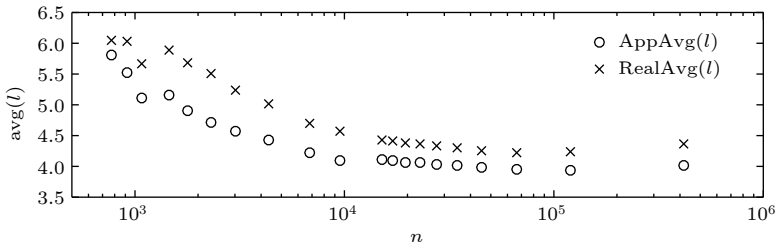

图 7 平均路径精确值、近似值与节点数的关系

Fig. 7. Relationships of RealAvg(l) and AppAvg(l) to n.

根据随机网络平均最短路径长度的估算公式计算相同规模随机网络的平均最短路径为 $L_{\text{random}} \sim \ln(N)/\ln(k) = 11.38$, 其中 N 是最大子网的节点数, $k = 4.274$ 是节点的平均度值, 根据互联网平均最短路径长度 [16] 公式, 可以计算得 $\langle i \rangle \sim 0.35 + 2.06 \times \lg N = 15.14$, 可知概念图谱网络的节点间的联系比同等规模的随机网络和万维网节点间的联系更紧密. 此外, 不同于互联网平均最短路径随网络规模的增大而增大, 概念图谱网络的平均路径随网络规模的增大反而减小, 并最终趋近于一个 4 左右的定值. 这一现象可能与概念图谱的结构有关, 为解释此现象, 对由算法 2 计算各阈值网络最大子网时得到的网络层次结构进行了统计. 表 8 是各阈值网络由中心层扩展时形成的层次结构及各层包含的节点数.

从表 8 可以看出, 这些阈值网络与概念图谱最大子网在结构上十分相似, 都形成了类似“菱形”结构: 大量节点集中在中间靠前的网络层, 少量节点处于两端的“边缘”层. 随着网络规模的增加, 大量的节点更是集中在了前 4 层, 表明大部分节点间经由中心层节点经过不超过 4 步就可到达彼此, 可以一定程度地解释概念图谱网络平均最短路径随着网络规模增加而趋近于 4 左右这个定值的原因. 概念图谱反映的是知识间的联系, 可以将其看作人拥有的知识. 通常一个人掌握的知识越多, 其由一个知识推理或搜索到另外一个知识的速度也就越快.

4) 聚类系数: 聚类系数 C 是所有节点聚类系数的平均值, 描述网络中节点的聚集情况, 即网络紧密性.

对于度为 1 的节点, 计算其聚类系数没有实际意义, 为此在计算网络的聚类系数前首先要剔除度值为 1 的节点. 在剔除度为 1 的节点后, 最大子网

表 8 不同阈值网络的层次结构及各层的节点数

Table 8. Layer structure and node number in each layer.

Layer	$t=10$	$t=20$	$t=30$	$t=50$	$t=100$	$t=200$	$t=300$	$t=500$	$t=1000$
1	2	2	2	2	2	2	2	2	2
2	26993	10212	6226	3409	1534	748	456	258	88
3	223659	65471	34244	16456	6445	2437	1305	558	103
4	131967	36095	21646	12077	5712	2829	1722	832	205
5	28219	6590	3563	2037	1116	656	661	438	108
6	3745	821	505	418	206	129	157	187	128
7	766	143	74	55	62	41	30	29	89
8	98	25	12	11	7	4	3	13	29
9	27	6	—	2	2	3	—	1	13
10	12	2	—	—	—	1	—	—	5
11	3	—	—	—	—	—	—	—	—

还有 5010477 个节点. 由于我们只有记录最大子网边信息的三元组的文本文件 GraphLink, 如果对每个节点直接计算其聚类系数, 首先需要遍历最大子网边集合 GraphLink 得到该节点的邻居节点集合, 为此需调用 3.3 节中的 GetNeighbor(Node, Graph) 函数, 并将参数 Graph 替换为 GraphLink; 之后再次遍历 GraphLink 得到邻居节点集合中存在的边的数量, 为此需再次调用 GetNeighbor(Node, Graph), 此处也应将 Graph 替换为 GraphLink. 参照 3.4 节中时间复杂度的分析, 可知计算一个节点聚类系数的时间复杂度为以上两个函数的时间复杂度之和, 即 $2n+3.2n=5.2n$, 此处的 n 为 GraphLink 包含的边数 (32274033). 由于需要计算聚类系数的节点数量为 5010477, 约为边数的 1/6, 所以计算网络的聚类系数的时间复杂度为 $5.2n\times n\times 1/6\approx 0.867n^2$, 时间复杂度仍较高, 所以我们采用分段式计算方法.

Node_{ki} 表示度值为 k_i 的节点的集合. 根据度分布可知, 度值越小, 对应的节点数量越大, 设定阈值 $\theta=100$.

对于度值大于 θ 的节点的集合 $\text{Node}_{ki}(ki=\theta{+}1, \theta{+}2, \cdots, k_{\max}$, 其中 $k_{\max}$ 表示节点的最大度值): 先根据最大连通子网的边集合 GraphLink 寻找到每个节点 $\text{Node}_j\in\text{Node}_{ki}$ 的邻居节点的集合 Neighbor_j^{ki} 然后遍历 GraphLink 中的每一条边, 对 Node_{ki} 中的每个 Node_j, 判断该边的两个端点是否都属于 Neighbor_j^{ki}, 若是, 则表示该边是连接 Node_j 的两个邻居节点的边. 度值为 k_i 的各个节点的邻居节点间的边数构成边数集合 Edge_{ki}, 其存储形式为 k-value 形式, 如 $\text{Edge}_{ki}=\{\{\text{Node}_1{:}\text{edge}_1\}, \{\text{Node}_2{:}\text{edge}_2\}, \cdots, \{\text{Node}_j{:}\text{edge}_j\}\}$, 其中 edge_j 为节点 Node_j 的邻居节点间的边数. 节点 Node_j 的聚类系数计算公式如下:

$$C_{\text{Node}\,j}=\frac{\text{Edge}_{ki}\,(\text{Node}_j)}{k_i\,(k_i-1)}, \tag{7}$$

其中 $\text{Edge}_{ki}(\text{Node}_j)$ 表示 Edge_{ki} 中与 Node_j 对应的 edge_j.

对于度值小于等于 θ 的节点集合 $\text{Node}_{ki}(ki=1, 2, \cdots, \theta)$: 首先同上得到与度值 k_i 对应的 Node_{ki} 及 $\text{Neighbor}_j^{ki}(j=1, 2, \cdots, \text{length}(\text{Node}_{ki}))$, 其中 $\text{length}(\text{Node}_{ki})$ 表示 Node_{ki} 中节点的数量; 然后根据 $(\text{Node}_{ki}\cup\text{Neighbor}_j^{ki}, j=1,2,\cdots,\text{length}(\text{Node}_{ki}))$ 从 GraphLink 中筛选与这些节点相关的边, 组成部分边集合 PartOfGraph_{ki}; 再对 PartOfGraph_{ki} 中的每一条边, 对 Node_{ki} 的每个 Node_j, 判断边的两个端点是否属于 Neighbor_j^{ki}, 若是, 则表示该边是连接 Node_j 的两个邻居节点间的边. 得到度值 k_i 对应的 Node_{ki} 中各个节点 Node_j 的邻居节点间的边数构成集合 Edge_{ki}, 然后根据 (7) 式计算该集合中每个节点的聚类系数.

由于度值越小, 节点数量越多, 聚类系数就越难计算. 在实验中, 度值在 2—10 范围内的节点数占所有可计算聚类系数节点的 89.66%, 所以我们在度值小于阈值时引入 Map/Reduce 模式, 将大型计算任务分解为多个小计算量任务, 然后同时进行计算. 对于度值相同的节点统一计算其聚类系数, 在计算时分别计算分子, 分母只需要计算一次 (分母是相同的). 并且, 此种计算模式在分析度-平

均聚类系数时也十分方便.

最大连通子网中聚类系数不为 0 的节点为 1837431 个, 占 12.16%, 由全部节点聚类系数计算得到的网络聚类系数为 0.0468; 剔除度值为 1 的节点后计算得到的网络聚类系数为 0.1410. 为了判断网络的小世界特性, 计算了相同规模 (节点数量和平均度相同) 的随机网络的聚类系数 $C_{\rm random}\sim k/N=2.83\times10^{-7}$, 其中 $N=15114834$ 为节点数, $k=4.274$ 是网络的平均度值[3]. 显然概念图谱网络的聚类系数远大于 $C_{\rm random}$, 可知概念图谱网络中节点聚群现象比较明显. 图 8 是度值与平均聚类系数的关系, 可知低度值节点的聚类系数较大, 高度值节点的聚类系数普遍较小.

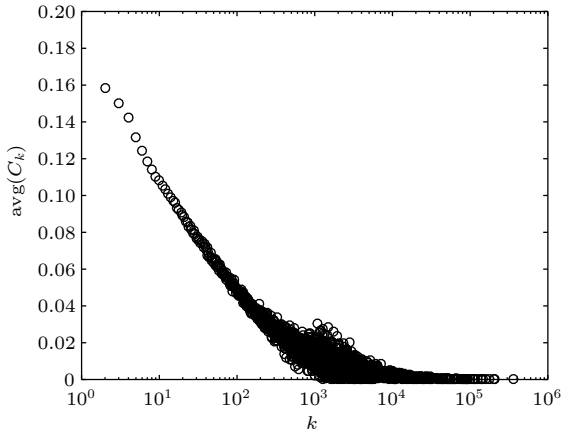

图 8 度值对应的平均聚类系数分布

Fig. 8. Average clustering coefficient distribution corresponding to degree.

5) 度相关性: 度相关性描述的是节点之间根据度值作为相互之间连接的选择偏好性, 如度值为 k 的节点的邻点平均度 $k_{nn}(k)$ 随 k 增加, 表示度大的节点偏好连接其他度大的节点, 则网络是正相关的; 反之, 如果 $k_{nn}(k)$ 随 k 而减小, 表示度大的节点偏好连接度小的节点, 则网络是负相关的.

表 9 列出了部分度值和该度值对应的所有节点的邻点平均度, 可知值最低的 5 个度其所有节点的邻点平均度非常大, 而值最高的 5 个度其所有节点的邻点平均度相对而言却非常小.

将度值 k 和与度为 k 的所有节点的邻点平均度 k_{nn} 的关系绘制成图 9, 可以看出, k_{nn} 随着 k 的增大而减小, 呈现负相关性, 表明概念图谱网中与高度值节点相连接的节点的度值偏低, 与低度值节点相连接的节点的度值偏高.

Newman[34] 还给出了一种通过网络节点度的 Pearson 相关系数 r 来判断网络相关性的量化方法, 具体公式如下:

$$r=\frac{M^{-1}\sum_i j_ik_i-\left[M^{-1}\sum_i\frac{1}{2}(j_i+k_i)\right]^2}{M^{-1}\sum_i\frac{1}{2}(j_i^2+k_i^2)-\left[M^{-1}\sum_i\frac{1}{2}(j_i+k_i)\right]^2},\tag{8}$$

其中 M 表示网络的边数, j_i 和 k_i 分别是第 i 条边 $(i=1,2,\cdots,M)$ 的两个端点的度值, r $(-1\leqslant r\leqslant1)$ 表示网络相关性. 经计算可知概念图谱最大连通子网的相关性为–0.067, 网络是负相关的, 与度值的邻点平均度分析结论相同. 同时, 网络的负相关性也保证了低度值节点可以与高度值节点相连, 由于高度值节点可以连接到很多其他节点, 所以在应用中可以实现推理过程.

6) 知识丢失对概念图谱完整性的影响: 用节点删除模拟知识丢失, 通过随机删除和定向删除 (度值由高到低) 分别测试了不同阈值下较小规模概念图谱网络的完整性. 图 10(a) 是阈值为 500 的实验结果, x 轴是节点删除比, y 轴是最大子网节点比例 S, 能够一定程度上描述概念图谱的完整性. 随机删除对 S 影响很小, 删除 80% 以上的节点时 S 才接近 0; 定向删除对 S 影响显著, 仅减少 0.5% 左右的节点时 S 就减少到了 0. 定向删除下 S 的下降规律与 computer network 十分相似, 快

表 9 部分度的节点数与该度值的所有节点的邻点平均度

Table 9. Part of N_k and $k_{nn}(k)$.

k	N_k	$k_{nn}(k)$	k	N_k	$k_{nn}(k)$
1	10104357	31235.02	159164	1(item)	122.577
2	2121672	13384.27	174283	1(product)	98.812
3	998490	10435.94	202331	1(issue)	91.266
4	523601	10231.12	204130	1(feature)	85.926
5	302171	10388.98	364343	1(factor)	56.088
⋮	⋮	⋮	⋮	⋮	⋮

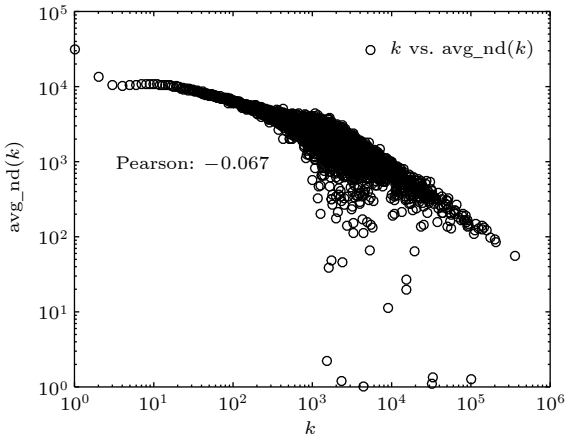

图 9 度-邻点平均度相关性分析

Fig. 9. Analysis of degree and average degree of neighbor nodes.

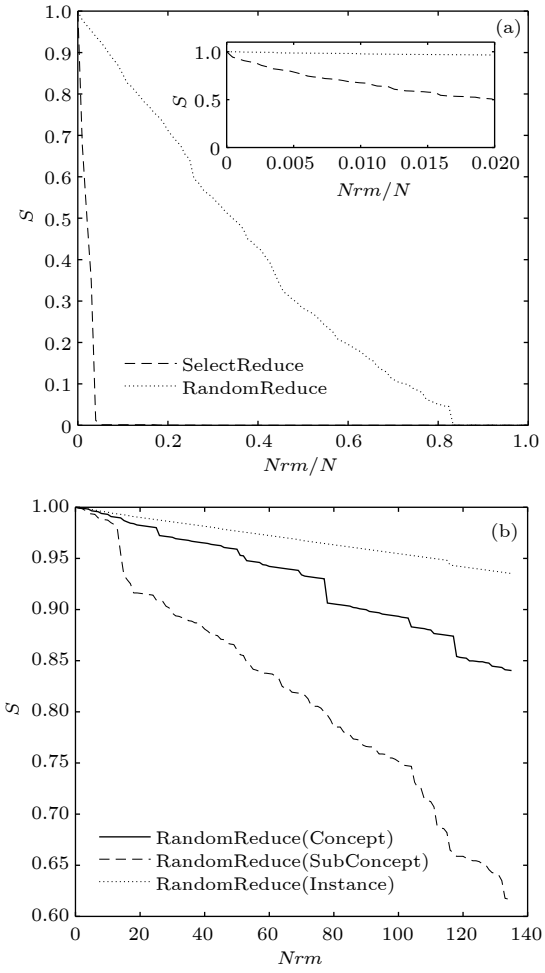

图 10 知识丢失对概念图谱完整性的影响

Fig. 10. Size of the giant component when nodes are removed.

于同是无尺度网的 BA 网和科学合作网[35].

还测试了概念、实例、子概念丢失对概念图谱完整性的影响, 图 10(b) 是随机删除 1—140 个三类节点时 S 的变化, 可见子概念的丢失对图谱完整性影响最大, 其次是概念, 最后是实例. 为了分析以上现象, 由度分布计算了各类节点的度期望值: 概念节点的度期望为 2.911, 实例为 1.899, 子概念为 36.214. 也就是说随机删除 1 个概念同时丢失约 3 条边 (知识之间的联系); 随机删除 1 个实例同时丢失不到 2 条边, 随机删除 1 个子概念, 平均会减少约 36 条边, 因此子概念的丢失对网络完整性影响最大.

概念图谱来源于数亿用户的搜索记录, 是反映人类对事物认知的知识库. 尽管每个人拥有的知识只是其中的一部分, 但却有类似的结构特征. 即最具体的实例对于掌握知识而言重要性最低, 而抽象的概念或子概念更重要. 现实世界中, 若忘记了某个实例, 比如忘记了 3 是 prime number (素数), 只要掌握了 prime number 的概念, 就可以推断出 3 是 prime number. 但如果忘记的是概念或者子概念, 如 prime number, 由 3 推理出 prime number 或其他素数就非常困难.

5 结 论

本文运用复杂网络理论对由微软概念图谱所构建的概念图谱网络进行了分析. 由于概念图谱网络中包含多个子网结构, 提出了一种适合概念图谱的最大子网抽取算法, 实验表明对于节点数量庞大的概念图谱网络, 该算法在时间和空间上都优于 NetworkX. 在进行节点度分布分析时不但发现最大连通子网具有无标度特性, 还发现子概念在概念图谱中扮演着连接其他节点的角色. 82% 的实例节点只与一个概念存在 IsA 关联, 向我们揭示了实例词在文本分析中通常不会因为一词多义的原因导致理解上的歧义. 为解决网络规模巨大导致的计算困难, 提出了一种近似网络平均距离的计算方法, 对比 NetworkX 和 CDZ 具有明显优势. 分析表明概念图谱网络具有小世界特性, 平均路径随网络规模增加而减小并趋于定值 4; 概念图谱网络的“菱形”结构揭示了平均路径趋于定值 4 的根源; 平均路径随网络规模增加而减小这一现象与人类认知和推理能力随知识增加而提升这一现象一致. 网络聚类系数较大, 网络中节点的群聚现象较为明显. 根据度相关性的分析, 可知网络中与高度值节点相连接的节点的度值偏低, 与低度值节点相连接的节

点的度值偏高, 度-平均度呈现负相关, 有利于实现概念图谱中的知识推理; 概念图谱完整性对知识的随机缺失不敏感且子概念对概念图谱完整性的影响明显高于概念和实例.

考虑到概念图谱的海量数据, 以上对全网特性的分析都没有考虑关系的方向和关系的紧密程度(边的长度), 在以后的工作中可以将关系的方向和边长引入概念图谱网络模型的构建中, 在此基础上利用局部拓扑特性进行概念图谱的自动补全研究.

参考文献

[1] Wang Z Y, Wang H X, Wen J R, Xiao Y H 2015 *ACM International Conference on Information and Knowledge Management* Melbourne, Australia, October 18–23, 2015 p653

[2] Hagberg A A, Schult D A, Swart P J 2008*Proceedings of the 7th Python in Science Conference* Pasadena, CA USA, August 19–24, 2008 p11

[3] Watts D J, Strogatz S H 1998 *Nature* **393** 440

[4] Barabási A L, Albert R 1999 *Science* **286** 509

[5] Liu Z H, Zeng Y, Wu H L, Ma J F 2014 *Journal of Computer Research and Development* **51** 2788 (in Chinese) [刘志宏, 曾勇, 吴宏亮, 马建峰 2014 计算机研究与发展 **51** 2788]

[6] An H Z, Zhong W Q, Chen Y R, Li H J, Gao X Y 2014 *Energy* **74** 254

[7] Almog A, Squartini T, Garlaschelli D 2015 *New J. Phys.* **17** 013009

[8] Xing X, Yu D X, Tian X J, Wang S G 2017 *Acta Phys. Sin.* **66** 230501 (in Chinese) [邢雪, 于德新, 田秀娟, 王世广 2017 物理学报 **66** 230501]

[9] Colombo A, Campos G R D, Rossa F D 2017 *IEEE Trans. Autom. Control* **62** 4933

[10] Wan X, Li Q M, Yuan J F, Schonfeld P M 2015 *Accid. Anal. Prev.* **82** 90

[11] Ye K H, Yuan X 2018 *Resources Development & Market* **34** 59 (in Chinese) [叶堃晖, 袁欣 2018 资源开发与市场 **34** 59]

[12] Hu Y H, Zhu D L 2009 *Physica A* **388** 2061

[13] Pien K C, Han K, Shang W L, Majumdar A 2015 *Transportmetrica A* **11** 772

[14] Albert R, Jeong H, Barabási A L 2000 *Nature* **406** 378

[15] Broder A, Kumar R, Maghoul F, Raghavan P, Rajagopalan S, Stata R, Tomkins A, Wiener J 2000 *Comput. Netw.* **33** 309

[16] Albert R, Jeong H, Barabási A L 1999 *Nature* **401** 130

[17] Aiello W, Chung F, Lu L Y 2001 *42nd Annual Symposium on Foundations of Computer Science* Las Vegas, NV, USA October 14–17, 2001 p510

[18] Aiello W, Chung F, Lu L Y 2000 *Proceedings of the Thirty-second Annual ACM Symposium on Theory of Computing* Portland, Oregon, USA, May 21–23, 2000 p171

[19] Rattigan M, Maier M, Jensen D 2006 *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* Philadelphia, PA, USA, August 20–23, 2006 p357

[20] Fiedor P 2015 *Acta Phys. Pol. A* **127** 863

[21] Wang G J, Xie C, Stanley H E 2018 *Comput. Econ.* **51** 607

[22] Qiu L, Jia T M, Yang H J 2016 *Acta Phys. Sin.* **65** 198901 (in Chinese) [邱路, 贾天明, 杨会杰 2016 物理学报 **65** 198901]

[23] Sun Y F, Wang C Y 2018 *Acta Phys. Sin.* **67** 148901 (in Chinese) [孙延风, 王朝勇 2018 物理学报 **67** 148901]

[24] Ruan Y R, Lao S Y, Wang J D, Bai L, Chen L D 2017 *Acta Phys. Sin.* **66** 038902 (in Chinese) [阮逸润, 老松杨, 王俊德, 白亮, 陈立栋 2017 物理学报 **66** 038902]

[25] Kitsak M, Gallos L K, Havlin S, Liljeros F, Muchnik L, Stanley H E, Makse H A 2010 *Nat. Phys.* **6** 888

[26] Li Q, Zhou T, Lu L Y, Chen D B 2014 *Physica A* **404** 47

[27] Ruan Y R, Lao S Y, Xiao Y D, Wang J D, Bai L 2016 *Chin. Phys. Lett.* **33** 028901

[28] Kong J T, Huang J, Gong J X, Li E Y 2018 *Acta Phys. Sin.* **67** 098901 (in Chinese) [孔江涛, 黄健, 龚建兴, 李尔玉 2018 物理学报 **67** 098901]

[29] Han D D, Yao Q Q, Chen Q, Qian J H 2017 *Acta Phys. Sin.* **66** 248901 (in Chinese) [韩定定, 姚青青, 陈趣, 钱江海 2017 物理学报 **66** 248901]

[30] Niu R W, Pan G J 2016 *Chin. Phys. Lett.* **33** 068901

[31] Jiang J, Zhang R, Guo L, Li W, Cai X 2016 *Chin. Phys. Lett.* **33** 108901

[32] Tang J Y, Wang T, Wang W 2011 *Journal of Software* **22** 2279 (in Chinese) [唐晋韬, 王挺, 王戟 2011 软件学报 **22** 2279]

[33] NetworkX Developers https://networkx.github.io/documentation/networkx1.9/modules/networkx/algorithms/components/connected.html#connected_component_subgraphs [2014-6-21]

[34] Newman M E J 2003 *Phys. Rev. E* **67** 026126

[35] Holme P, Kim B J, Yoon C N, Yoon C N, Han S K 2002 *Phys. Rev. E* **65** 056109

Empirical study of knowledge network based on complex network theory*

Ding Lian-Hong [1)] Sun Bin [1)] Shi Peng [2)†]

1) (*School of Information, Beijing Wuzi University, Beijing* 101149, *China*)

2) (*National Center for Materials Service Safety, University of Science and Technology Beijing, Beijing* 100083, *China*)

(Received 20 January 2019; revised manuscript received 21 April 2019)

Abstract

Knowledge graph is a hot topic in artificial intelligence area and has been widely adopted in intelligent search and question-and-answer system. Knowledge graph can be regarded as a complex network system and analyzed by complex network theory, which studies the interaction or relationship between various factors and basic characteristics of complex system. Its characteristics and their physical meanings are very helpful in understanding the nature of the knowledge graph. Concept graph is a large-scaled knowledge graph published by Microsoft. In this paper, we construct a huge complex network according to Microsoft's concept graph. Its complex network characteristics, such as degree distribution, average shortest distance, clustering coefficient and degree correlation, are calculated and analyzed. The concept graph is not a connected network and its scale is very large; an approach is proposed to extract its largest connected subnet. The method has obvious advantages in both time complexity and space complexity. In this paper, we also present a method of calculating the approximate average shortest path of the largest connected subnet. The method estimates the maximum and minimum value of the shortest distance between nodes according to the distance between the central node and the network layer that the node belongs to and the distance between different layers. In order to calculate the clustering coefficient, different methods are introduced for nodes with different degree values and Map/Reduce idea is adopted to reduce the time cost. The experimental results show that the largest subnet of the concept graph is an ultra-small world network with the characteristics of scale-free. The average shortest path length decreases towards 4 with the network size increasing, which can be easily explained by the diamond-shaped network structure. The concept graph is a disassortative network where low degree nodes tend to connect to high degree nodes. The subConcepts account for 99.5% of nodes in the innermost k-core after k-shell decomposition. It shows that the subConcepts play an important role in the connectivity of network. The absence of subConcept affects the complexness of concept graph most, the concept next, and the instance least. The 82% instance nodes and 40% concept nodes of the concept graph each have a degree value of 1. It is believed that compared with the concept words, the instance words do not lead to the ambiguity in the understanding of natural language, caused by polysemy.

Keywords: microsoft concept graph, complex network, subnet extraction, approximate average path

PACS: 89.75.Fb, 05.90.+m, 02.10.Ox **DOI:** 10.7498/aps.68.20190106

* Project supported by the National Key R&D Program of China (Grant No. 2017YFB0203703), the Key Program of the National Natural Science Foundation of China (Grant No. 71831001), the Science and Technology Plan General Program of Beijing Municipal Education Commission, China (Grant No. KM201910037186), the Social Science Grant of Beijing Municipal Education Commission, China (Grant No. SM201610037001), and the Research Team Collaborative Fund of Beijing Wuzi University, China (Grant No. 2017GG05).

† Corresponding author. E-mail: shipengustb@sina.com