基于改进的 TextRank 的自动摘要提取方法

余珊珊[1] 苏锦钿[2] 李鹏飞[2]
(广东药科大学医药信息工程学院 广州 510006)[1] (华南理工大学计算机科学与工程学院 广州 510640)[2]

摘 要 经典的 TextRank 算法在文档的自动摘要提取时往往只考虑了句子节点间的相似性,而忽略了文档的篇章结构及句子的上下文信息。针对这些问题,结合中文文本的结构特点,提出一种改进后的 iTextRank 算法,通过将标题、段落、特殊句子、句子位置和长度等信息引入到 TextRank 网络图的构造中,给出改进后的句子相似度计算方法及权重调整因子,并将其应用于中文文本的自动摘要提取,同时分析了算法的时间复杂度。最后,实验证明 iTextRank 比经典的 TextRank 方法具有更高的准确率和更低的召回率。

关键词 中文文本,自动摘要提取,TextRank,篇章结构,无监督学习方法
中图法分类号 TP181 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2016.6.048

Improved TextRank-based Method for Automatic Summarization

YU Shan-shan[1] SU Jin-dian[2] LI Peng-fei[2]
(College of Medical Information Engineering, Guangdong Pharmaceutical University, Guangzhou 510006, China)[1]
(College of Computer Science and Engineering, South China University of Technology, Guangzhou 510640, China)[2]

Abstract The canonical TextRank usually only considers the similarity between sentences in the processes of automatic summarization and neglects the information of text structures and sentence contexts. To overcome these disadvantages, we proposed an improved method on the basis of TextRank, called iTextRank, by incorporating the structure information of Chinese texts. iTextRank takes some important contexts and semantic information into consideration, including titles, paragraphs, special sentences, positions and lengths of sentences, when building the network diagram of TextRank, computing the similarities of sentences and adjusting the weights of the nodes. We also applied iTextRank into the automatic summarization of Chinese texts and analyzed its time complexities. Finally, some experiments were done. The results prove that iTextRank has higher accuracy rate and lower recall rate compared with canonical TextRank.

Keywords Chinese texts, Automatic summarization extraction, TextRank, Article discourse, Unsupervised learning methods

1 引言

近年来,不少学者针对汉语的特点对中文文本的自动摘要提取展开了相关的研究。例如,文献[1]提出一种基于篇章结构的中文 Web 文档自动摘要技术,主要通过篇章结构分析、词语权重计算、关键词提取并统计句子的权重等方式生成摘要,但没有考虑句子间的关系。文献[2]提出一种基于回归模型的句子相似度计算方法,重点考虑了词语的前后位置信息,但忽略句子与标题的关系、句子的位置以及特殊句子等信息。文献[3]提出一种基于 LexRank 改进算法的自动摘要系统,考虑了指示性词语特征、句子长度及位置信息,但没有考虑标题和特殊句子等。文献[4,5]采用有监督的机器学习方法,并将熵和相关度等性质作为句子的特征来选择文档的摘要。其中,文献[4]主要基于线性回归和 ELM 回归(Extreme Learning Machine),而文献[5]主要基于 LDA(Latent Dirichlet Allocation)模型和主题模型。这两者均采用有监督的学习方法,准确率较高,但容易受训练样本的影响,而且领域通用性较差,不适合用于海量文本的摘要提取。

TextRank 算法[6]是 Mihalcea 和 Tarau 于 2004 年在研究自动摘要提取过程所提出来的,主要是借鉴 Google 公司 PageRank 算法的思路,将句子间的相似关系看成是一种推荐或投票关系,由此构建 TextRank 网络图,并通过迭代计算至收敛来得到句子的权重值。在此基础上,文献[7-9]将 TextRank 应用于信息的检索,其中文献[7,8]根据一定窗口内词项的共现信息构建无权的 TextRank 网络图,而文献[9]则进一步利用词项间的共现频率作为边的权重来构建加权网络。文献[10-12]将 TextRank 应用于关键词的提取,其中文献[10]主要考虑了词与词之间、句子与句子之间、词与句子之间等文章结构信息,文献[11]主要考虑词频、词性和词语间的语义关系等信息,而文献[12]则通过引入社会化标签 Tag 的方式来调整 TextRank 词项图中边的权重,并用于计算词项的重要度。上述研究主要是在词频的基础上利用各种关联度计算方法(如互信息、Pearson's χ^2 统计量、Dice 系数等)计算词项间的关联度,并构建相应的无权或加权 TextRank 网络图,不适合直

到稿日期:2016-01-20 返修日期:2016-03-20 本文受广东省自然科学基金(2015A030310318),广东省医学科学技术研究基金项目(A2015065),国家自然科学基金资助项目(61103038)资助。
余珊珊(1980-),女,博士,讲师,主要研究方向为本体、范畴论、程序语言、自然语言理解,E-mail:susyu@139.com;苏锦钿(1980-),男,博士,副教授,主要研究方向为大数据、形式语义和范畴论;李鹏飞(1993-),男,主要研究方向为文本挖掘。

接应用于句子，也没有考虑标题、段落、特殊句子、句子位置和长度等信息。而这些信息在有监督的学习方法中往往起到重要的作用。

本文在上述研究的基础上结合篇章结构及句子的上下文信息对经典的 TextRank 算法进行改进，并将研究结果应用于中文文本的自动摘要提取。本文第 2 节介绍经典的TextRank 算法；第 3 节构建以句子为单位的加权 TextRank 网络图；第 4 节提出改进后的 iTextRank 算法；第 5 节通过实验对 iTextRank 进行验证和分析；最后是总结并给出下一步的工作方向。

2 经典的 TextRank 算法

经典的 TextRank 算法是在 Google 公司 PageRank 算法的启发下，利用投票的原理让每一个节点为它的邻居节点投赞成票，票的权重取决于节点本身的票数。这是一个“先有鸡还是先有蛋”的悖论。TextRank 算法借鉴了 PageRank 的做法，采用矩阵迭代收敛的方式解决了这个悖论。在 TextRank 算法中，由节点及节点间的链接关系可构成一个无向的网络图。

假设 $V=\{V_1,V_2,\cdots,V_n\}$ 是由 n 个元素 $V_i(1\leqslant i\leqslant n)$ 所构成的集合。以 V 中各个 V_i 为节点并以节点间的相似关系为边可构成一个无向的 TextRank 网络图 $G=(V,E,W)$，其中 $E\subseteq V\times V$ 为节点间各个边的非空有限集合，记为 $E=\{(V_i,V_j)\mid V_i\in V\wedge V_j\in V\wedge w_{ij}\in W\wedge w_{ij}\neq 0\}$；$W=\{w_{ij}\mid 1\leqslant i\leqslant n\wedge 1\leqslant j\leqslant n\}$ 为边上的权重集合，w_{ij} 为节点 V_i 与 V_j 间边的权重值，可通过各种距离相似度计算函数（如欧氏距离、Jaccard 或余弦函数等）计算所得。

由图 $G=(V,E,W)$ 可得到节点间的一个 $n\times n$ 的相似度矩阵 $SD_{n\times n}$：

$$SD_{n\times n}=\begin{pmatrix} w_{11} & \cdots & w_{1n} \\ \vdots & \ddots & \vdots \\ w_{n1} & \cdots & w_{nn} \end{pmatrix} \tag{1}$$

显然，$SD_{n\times n}$ 为一个对称矩阵，即节点 V_i 对 V_j 的贡献度和 V_j 对 V_i 的贡献度是一样的，其对角线上元素的值均为 1。

根据给定的 G 及相应的相似度矩阵 $SD_{n\times n}$ 可迭代计算各个节点的权重，具体的权重计算如式(2)所示：

$$WS(V_i)=\frac{1-d}{n}+d*\sum_{V_j\in In(V_i)}\frac{w_{ij}}{\sum_{V_k\in Out(V_j)}w_{jk}}WS(V_j) \tag{2}$$

其中，$WS(V_i)$ 是节点 V_i 的权重值（称为 PR 值），d 是阻尼系数，一般设置为 0.85。$In(V_i)$ 代表指向 V_i 的节点集合，$Out(V_i)$ 代表 V_i 所指向的节点集合，$|Out(V_j)|$ 是集合 $Out(V_j)$ 中元素的个数。

式(2)的左边表示 V_i 的权重，右侧的求和表示每个相邻节点对本节点的贡献程度。求和的分子 w_{ij} 表示两个节点 V_i 和 V_j 间的相似程度，分母为一个加权和，$WS(V_j)$ 代表上一次迭代后节点 V_j 的权重值。

由于计算节点的权重时又需要用到节点本身的权重，因此需要进行迭代计算。假设每个节点的权重初始值均为 $1/|V|$，即 $B_0=(1/|V|,\cdots,1/|V|)^{\mathrm{T}}$，则一般经过若干次迭代计算后可收敛：

$$B_i=SD_{n\times n}\cdot B_{i-1} \tag{3}$$

PageRank 算法中证明了式(3)中的迭代计算方式在满足一些合适的条件时最终会收敛。因此，当两次迭代的结果 B_i 和 B_{i-1} 差别非常小并接近于零时停止迭代计算，此时算法结束，并最终可得到包含各个节点权重值的向量。H. P. Luhn 在文献[13]中指出在阈值为 0.0001 的情况下，一般经过20～30 次就可以到达收敛。根据权重值的大小进行倒排序就可得到相应的排名。

3 文本的 TextRank 网络图构造

3.1 文本预处理及特征选择

给定一段中文文本，将每一个句子作为一个窗口进行分词可得到该句子的各个特征项。由于这些特征项的维数非常高，存在许多对摘要提取无用的特征，因此一方面利用停用词表去掉对这些无用的词，并进行敏感词的过滤；另一方面，根据 Zipf 法则（也称幂集定律）合理地删除大量的低频词，降低特征空间的维数。为减少冗余度和提高效果，还可以采取特征词相关性分析、聚类、同义词和近义词归并等策略。有些词语虽然字面表示不同，但含义相同。例如“计算机”、“电脑”、“Computer”、“PC 机”虽属于不同的词语，但却表达相同含义，在词频统计时应将其作为一个词条进行处理。图 1 给出中文文本的特征词预处理流程。

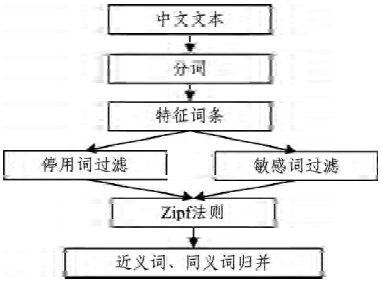

图 1 句子的特征词预处理流程

每一个句子经过图 1 的处理流程后可得到各自的特征词向量，接着利用相似度函数计算句子间的相似度，并将其作为句子间边的权重。若两个句子之间不存在相似度，则意味着它们之间不存在相应的边。

给定一段中文文本 D，假定包含 n 个句子，则 D 可表示为集合 $D=\{P_1,\cdots,P_n\}$，其中每一个 $P_i(1\leqslant i\leqslant n)$ 为按 D 中出现的先后顺序进行排序的句子。根据图 1 的预处理流程对 D 及各个 P_i 进行处理可得到：

(1) D 的特征词向量，记为 $D_{key}=[key_1:tn_1,\cdots,key_j:tn_j,\cdots,key_h:tn_h](1\leqslant j\leqslant h)$，其中 $h=|D_{key}|$ 为文本中所有特征词的数量，tn_j 为特征词 key_j 在 D 中出现的词频。

(2) 每一个 P_i 的一维向量 $P_{ih}=[key_{i1}:num_{i1},\cdots,key_{ij}:num_{ij},\cdots,key_{ih}:num_{ih}](1\leqslant j\leqslant h)$，其中 num_{ij} 为特征词 key_{ij} 所对应的词频。若 key_j 在 P_i 中出现，则相应的 num_{ij} 为 key_j 在该句子中的词频，否则 num_{ij} 为 0。

由所有的 P_{ih} 可构成一个矩阵 $D_{n\times h}$：

$$D_{n\times h}=\begin{pmatrix} k_{11} & \cdots & k_{1h} \\ \vdots & \ddots & \vdots \\ k_{n1} & \cdots & k_{nh} \end{pmatrix}$$

其中，第 i 行表示 P_{ih}，对应的每一列 k_{ij} 为 P_i 中对应的分量 $key_{ij}:num_{ij}$。显然，$D_{n\times h}$ 为一个高维稀疏矩阵。

在选择句子及文本的特征词时，本文采用基于单词频度（Term Frequency，TF）和逆文档频度（Inverted Document Frequency，IDF）的 TF-IDF 权值法作为特征词的评估函数。为了考虑文档长度对权值的影响，并弱化词频差异较大所带来的影响，下面在 TF-IDF 的基础上结合文档长度对权值进

行规范化，并用词频对数的形式代替 TF-IDF 中的词频。以 D_{key} 为例，其具体的特征词评估函数的定义如下：

$$WS(key_j)=\frac{\lg(tn_j+1)\times\lg\frac{N}{N_{key_j}}}{\sqrt{\sum_{j=1}^{h}(\lg(tn_j+1)\times\lg\frac{N}{N_{key_j}})^2}} \tag{4}$$

其中，N 为分词工具中词典所包含的特征词的总数，N_{key_j} 为 key_j 在 N 中出现的次数。

根据计算结果对特征词进行排序，并取前若干个特征词就可得到文档对应的关键词列表。在实际应用中可根据文本的长度调整相应的特征词数量阈值。

3.2 TextRank 网络图

给定各个句子的特征向量空间，可以通过各种距离相似度计算函数（如欧氏距离、Jaccard 或余弦函数等）计算句子间的相似度。这里采用余弦相似度函数：

$$w_{ij}=\frac{P_{ih}\cdot P_{jh}}{\|P_{ih}\|\cdot\|P_{jh}\|}=\frac{\sum_{1\leqslant m\leqslant h}WS(num_{im})\times WS(num_{jm})}{\sqrt{\sum_{1\leqslant p\leqslant h}WS(num_{ip})^2}\sqrt{\sum_{1\leqslant p\leqslant h}WS(num_{jp})^2}} \tag{5}$$

其中，· 为向量点积。

由矩阵 $D_{n\times h}$ 与式(5)可得到另一个矩阵 $SD_{n\times n}$：

$$SD_{n\times n}=D_{n\times h}D_{n\times h}^{\mathrm{T}}=\begin{pmatrix}w_{11}&\cdots&w_{1n}\\ \vdots&\ddots&\vdots\\ w_{n1}&\cdots&w_{nn}\end{pmatrix} \tag{6}$$

其中，$SD_{n\times n}$ 为句子间的相似度矩阵，权值 w_{ij} 表示句子 P_i 与 P_j 间的相似度。显然，$SD_{n\times n}$ 为一个对称矩阵，且其对角线上元素的值均为 1。

以 D 中各句子 P_i 为节点并以句子间的相似关系为边并以句子间的相似度为边的权值可构成一个无向的加权 TextRank 网络图，其中各节点的权重计算：

$$ws(P_i)=\frac{1-d}{n}+d*\sum_{P_j\in In(P_i)}\frac{w_{ij}}{\sum_{P_k\in Out(P_j)}w_{jk}}ws(P_j) \tag{7}$$

设每个节点的权重初始值均为 $1/|D|$，即 $B_0=[1/|D|,\cdots,1/|D|]^{\mathrm{T}}$，则经过若干次迭代计算后可收敛：

$$B_i=SD_{n\times n}\cdot B_{i-1} \tag{8}$$

收敛后的 B_i 包含了各个句子节点的权重值。根据值的大小进行倒排序可得到相应的句子重要性排名。再选择一定数量 $N(1\leqslant N\leqslant|D|)$ 的句子，并结合它们在文本中的先后顺序进行组织就可构成该文本的摘要。在实际应用中，可利用一些统计信息来确定 N 的值。例如，根据语料库统计分析摘要句子数量与文本句子数量间的比例以确定合适的 N 值，或者直接简单地取文本句子总数的一定比例作为摘要句子的数量。

例 1 表 1 是一段关于 PageRank 原理的介绍文本。以句号作为窗口结束的标志，整个文本共包含 10 句，每一段均为 5 句，按其先后顺序分别用 A1、A2、A3、A4、A5 和 B1、B2、B3、B4、B5 表示。

表 1 PageRank 介绍

标题	Google 公司的 PageRank 原理介绍
原文	通过对由超过 50000 万个变量和 20 亿个词汇组成的方程进行计算，PageRank 能够对网页的重要性做出客观评价。PageRank 并不计算直接链接的数量，它是将从网页 A 指向网页 B 的链接解释为由网页 A 对网页 B 所投的一票。这样，PageRank 会根据网页 B 所收到的投票数量来评估该网页的重要性。此外，PageRank还会评估每个投票网页的重要性，因为某些重要网页的投票被认为具有较高的价值，这样，它所链接的网页就能获得较高的价值。这就是 PageRank 的核心思想，当然 PageRank 算法的实际实现上要复杂很多。 但是问题又来了，计算其他网页 PageRank 的值需要用到网页本身的 PageRank 值，而其他网页的 PageRank 值反过来又影响本网页的 PageRank 的值，这不就成了一个先有鸡还是先有蛋的问题了吗？Google 的两个创始人拉里. 佩奇（Larry Page）和谢耳盖. 布林（Sergey Brin）把这个问题变成一个二维矩阵相乘的问题，并且用迭代的方法解决了这个问题。他们先假定所有网页的排名是相同的，并且根据这个初始值，算出各个网页的第一次迭代的排名，然后再根据第一次迭代排名算出第二次的排名。他们两人从理论上证明了不论初始值如何选取，这种算法都将能够保证网页排名的估计值能够收敛到它们就有的真实值。值得一提的是，这种算法的执行是完全没有任何人工干预的。
分词并进行预处理后	标题：Google/公司/PageRank/原理/介绍 正文：超过/50000 万/变量/20 亿/词汇/组成/方程/进行/计算/，/PageRank/能够/网页/重要性/做出/客观/评价/。PageRank/计算/直接/链接/数量/，网页/A/指向/网页/B/链接/解释/为由/网页/A /网页/B/投/票/。PageRank/网页/B/收到/投票/数量/评估/网页/A/重要性/。PageRank/评估/每个/投票/网页/重要性，/重要/网页/投票/认为/具有/高/价值/，链接/网页/获得/高/价值/PageRank/核心/思想/，PageRank/算法/实际/实现/复杂/。 问题/，计算/网页/PageRank/值/需要/网页/PageRank/值/，网页/PageRank/值/影响/网页/PageRank/值/，成/一个/先/鸡/先/蛋/问题/。Google/两/创始人/拉里/佩奇/LarryPage/谢耳/盖/布林/SergeyBrin/问题/变成/一个/维/矩阵/相乘/问题/，迭代/方法/解决/问题/。先/假定/所有/网页/排名/相同/，初始值/，算/网页/第一/次/迭代/排名/，再/第一/次/迭代/排名/。算/第二/次/排名/两/人/理论/证明/初始值/选取/，这种/算法/能够/保证/网页/排名/。估计/值/能够/收敛/真实/值/。值得一提的是/，这种/算法/执行/完全/人工/干预/。

由表 1 可得到一个 10×86 的矩阵 $D_{10\times86}$：

$$D_{10\times86}=\begin{pmatrix}A_{1\times1}&\cdots&A_{1\times86}\\ \vdots&\ddots&\vdots\\ A_{5\times1}&\cdots&A_{5\times86}\\ B_{1\times1}&\cdots&B_{1\times86}\\ \vdots&\ddots&\vdots\\ B_{5\times1}&\cdots&B_{5\times86}\end{pmatrix}=\begin{pmatrix}k_{1,1}&\cdots&k_{1,86}\\ \vdots&\ddots&\vdots\\ k_{10,1}&\cdots&k_{10,86}\end{pmatrix}$$

利用式(7)可得到相应的矩阵 $SD_{10\times10}$：

$$SD_{10\times10}=D_{10\times86}D_{10\times86}{}^{\mathrm{T}}=\begin{pmatrix}w_{1\times1}&\cdots&w_{1\times10}\\ \vdots&\ddots&\vdots\\ w_{10\times1}&\cdots&w_{10\times10}\end{pmatrix}=\begin{pmatrix}1.000&0.247&0.377&0.269&0.158&0.283&0.000&0.069&0.160&0.000\\ 0.247&1.000&0.595&0.502&0.104&0.435&0.000&0.182&0.140&0.000\\ 0.377&0.595&1.000&0.596&0.191&0.456&0.000&0.167&0.129&0.000\\ 0.269&0.502&0.596&1.000&0.114&0.362&0.000&0.149&0.115&0.000\\ 0.158&0.104&0.191&0.114&1.000&0.319&0.000&0.000&0.067&0.120\\ 0.283&0.435&0.456&0.362&0.319&1.000&0.161&0.175&0.322&0.000\\ 0.000&0.000&0.000&0.000&0.000&0.161&1.000&0.101&0.00039&0.000\\ 0.069&0.182&0.167&0.149&0.000&0.175&0.101&1.000&0.207&0.000\\ 0.160&0.140&0.129&0.115&0.067&0.322&0.039&0.207&1.000&0.161\\ 0.000&0.000&0.000&0.000&0.120&0.000&0.000&0.000&0.161&1.000\end{pmatrix}$$

图 2 以 A1 和 B1 为例给出它们与其它各节点的 TextRank 网络图。

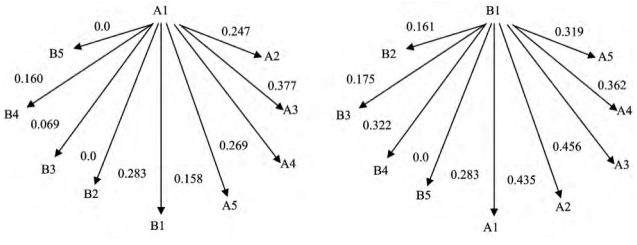

图 2 A1 和 B1 到其它节点的 TextRank 网络图

设各个句子的初始权重值为 $B_0=[1/10,\cdots,1/10]$。以 A1 为例根据式(7)对其权重进行第一次计算：

$$ws(A1)=\frac{(1-0.85)}{10}+0.85\times\sum_{P_j\in In(P_i)}\frac{w_{ij}}{\sum_{P_k\in Out(P_j)}w_{jk}}\times\frac{1}{10}$$

第一次迭代计算的结果为 $ws(A1)\approx0.086$。经过 12 次迭代可得到稳定值 $ws(A1)\approx0.101$。类似地，可计算得到其他节点的权重值。经过 16 次迭代计算至收敛，并得到最终矩阵：$B_{16}=[0.101,0.136,0.153,0.130,0.078,0.162,0.032,0.077,0.098,0.032]$。

取 3 个句子作为摘要句子(即第 2、3 和 6 句)，最终可得到摘要，如表 2 所列。

表 2 PageRank 介绍的摘要提取结果

摘要	PageRank 并不计算直接链接的数量，它是将从网页 A 指向网页 B 的链接解释为由网页 A 对网页 B 所投的一票。这样，PageRank会根据网页 B 所收到的投票数量来评估该网页的重要性。但是问题又来了，计算其他网页 PageRank 的值需要用到网页本身的 PageRank 值，而其他网页的 PageRank 值反过来又影响本网页的 PageRank 的值，这不就成了一个先有鸡还是先有蛋的问题了吗？

4 基于改进 TextRank 算法的自动摘要提取

4.1 改进 TextRank 算法

第 2 节中的经典 TextRank 算法只考虑句子间的相似度，而忽略以下信息。

(1)句子与标题之间的相似度

从文章结构来说，标题是文章的概括和总结，出现在标题中的词也很有可能是特征词。文章中的句子与标题的相似度越高，说明这个句子越能表达文章的主要内容，那么它被抽取成摘要的可能性越大。因此，可考虑通过计算句子与标题的相似度来调整该句的权重。为了区别正文中的句子 P_i，将标题句子记为 P_0，其特征词向量表示为 $P_0=[k_{01},\cdots,k_{0h'}]^{\mathrm{T}}$，这里的 h'是扩展后包含标题及句子的特征词的数量。

句子与标题的相似度可分两个方面考虑：

1)计算各个句子与标题句子间的相似度，相似度越高的则权重越高，反之则越低。

在 TextRank 算法中，经过多次迭代计算后每个句子的重要性趋于稳定，且这个稳定值与初始值无关，而只与其它句子对本句子的贡献度有关。这意味着简单地通过某种策略(比如对某些句子节点指定更多的权重)来分配初始权值以改变计算结果是无意义的。因此，可考虑利用标题 P_0 与各个句子 P_i 间的相似度对最终收敛后的 B_i 中的句子节点的权重进行调整。

若标题与句子的特征词完全相同，即其相似度为 1，则该句子的最终权重再放大 2 倍；若完全不同，则保持原权重。具体计算公式如下：

$$tws_i=1+sim(P_{ih'},P_{0h'})=1+\frac{P_{ih'}\cdot P_{0h'}}{\|P_{ih'}\|\cdot\|P_{0h'}\|}\qquad(9)$$

由式(9)可得到相应的向量 $TD_{n\times1}=[tws_1,\cdots,tws_n]^{\mathrm{T}}$。

2)各个句子中的特征词是否在标题中也同时出现。若特征词在标题出现，则提升其词频的权重；否则，保持其词频的权重不变。具体的计算公式如下。

$$\forall\,1\leqslant i\leqslant n.\ \forall\,1\leqslant j\leqslant h'.\ \text{if } P_0.k_{0j}\neq0\wedge P_i.k_{ij}\neq0 \text{ then } P_i.k_{ij}=P_i.k_{ij}\times1.2 \text{ else } P_i.k_{ij}=P_i.k_{ij}\qquad(10)$$

(2)特殊段落中的句子位置

美国 RE. Baxendale 的研究结果[14]显示：人工摘要中选取段首句作为摘要的比例为 85%，选取段尾句作为摘要的比例为 7%。美国康奈尔大学的 G. Salton 就是以此为思路提出了寻找文章的中心段落作为摘要核心的思想[15]。基于此，可根据段落及段落中句子的位置进行加权，对首段中越靠前的句子给予越大的权重提升，末段中越靠后的句子给予越小的权重提升。考虑到收敛后的权重矩阵 B_i 仍按句子的先后顺序排序，因此可根据句子的位置设置相应的权重调整向量。

假设首段中包含 s 个句子，末段包含 r 个句子，则对 B_i 中句子的权重调整可以通过转移矩阵 $FP_{n\times1}=[pw_1,\cdots,pw_s,\cdots,pw_{n-r+1},\cdots,pw_n]^{\mathrm{T}}$ 实现。$FP_{n\times1}$ 中前 s 个 $pw(1\leqslant j\leqslant s)$的值采用依次递减的方式，而后 r 个句子采用依次递增的方式。具体的计算公式为

$$\begin{cases}(1+e_1)-(j-1)\times\dfrac{e_1}{s}, & 1\leqslant j\leqslant s\\ 1+(j-n+r)\times\dfrac{e_2}{r}, & n+1-r\leqslant j\leqslant n\\ 1, & \text{else}\end{cases}\qquad(11)$$

其中，e_1 和 e_2 均为调整阈值。例如，假设 $e_1=e_2=0.2$，$s=r=3$，对于 P_1、P_2 和 P_3，计算结果分别为(取小数点后 3 位)：$pw_1=1.200$，$pw_2=1.133$，$pw_3=1.067$。类似地，可计算后 3 句并得到相应的值为：$pw_{n-2}=1.067$，$pw_{n-1}=1.133$，$pw_n=1.200$。最终可得到向量 $FP_{n\times1}=[1.200,\cdots,1.133,1.067,\cdots,1.067,1.133,1.200]^{\mathrm{T}}$。

通过矩阵相乘 $B_{i+1}=B_i\cdot FP_{n\times1}$ 就可根据句子的位置对最终的权重进行相应的调整。

(3)关键句子的处理

由于中文文章的特点是以句号为结束标志，因此如果一个句子自成一段，那么这个段落往往起着“承上启下”或者“过渡句”的作用。另外，文章中可能有一些小标题，也是自成一段的。这些段落一般具有高概括性、精炼性的特点，符合摘要本身的要求，故有更大的可能性作为摘要的一部分。

对此类句子可以给予更大的权重，处理方法类似于前面对首段和末段句子的权重提升。但这里需要对此类句子进行筛选，因为文章中往往也有一些无意义的短句子(一般字数小于或等于 6)自成一段，这些句子就没有提升权重的必要。具体计算过程如下。

①输入：句子集合 P=[P1，P2，…，Pn]，输出：Bi。

②计算：

```
for (int k=0;k<n;k++){
  if isSection(Pk) then { //判断 Pk 是否为段落
//句子字数大于 6，则更新关键句的权重
if count(Pk)> 6 then { Bi[k]=1.1×Bi[k];}}
```

(4)特殊句子权重的传递

在 TextRank 算法中，句子的重要程度是由句子本身所得到的其它句子的“投票”数量和质量决定的。对于特殊句子而言，其本身权重得到提升后，将更大的权重传递给与其相关的句子(即提升此相关句子得到的投票质量)，则会使最终的计算结果更加精准。因此，对首段句子、末段句子、关键句子(以下统称为特殊句子)进行标记，并放大这些句子传递出去的权重，使与之关联的句子获得更大的权值。

由式(7)知 w_{ij} 既表示句子 P_i 和 P_j 的相似度，同时也表示 P_i 给予 P_j 的权重大小。对于特殊句子 P_i，可根据特殊句子的重要程度确定一个大于 1 的系数 a 用于放大相似度性矩阵 $SD_{n\times n}$ 中第 i 行的值。在 TextRank 算法中，句子两两之间都是有关系的，因此对所有句子放大同样的权重是无意义的。这里取一个阈值 0.25 作为界限来放大权重，即使得原本与特殊句子有更大相似性的句子获得更大的权值。具体计算过程如下。

①输入：数组 key[L]，用于记录特殊句子的位置。

输出：$SD_{n\times n}$。

②计算：

```
for (int i=0;i<L;i++){
  for (int j=0;j<n;j++){//处理矩阵 SDn×n的第 i 行
  //若原相似度大于 0.25，则放大 a 倍
    if SDn×n[i,j]>0.25 {SDn×n[i,j]=SDn×n[i,j]×a;}}}
```

(5)句子长度过滤

考虑一个句子是否作为摘要候选句时，该句子的长度也应作为一个考虑条件，过长或过短的句子都不应该作为生成摘要的候选句。例如，对于经过预处理后不包含基本特征词的句子就可以直接忽略。

4.2 算法实现

结合图 3，iTextRank 的具体实现及计算过程如下。

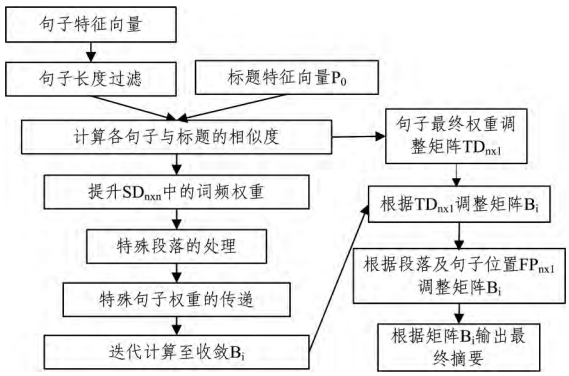

图 3 iTextRank 算法的基本流程

1. 根据文档 D 构造各个句子的特征词向量 P_i，并进行句子的长度过滤。

2. 得到特征词词频矩阵 $D_{n\times h'}$。

3. 得到标题的向量矩阵 $P_0=[k_{01},\cdots,k_{0h'}]^{\mathrm{T}}$，并分别计算各句子与标题的相似度。

3.1 由式(9)得到向量 $TD_{n\times 1}=[tws_1,\cdots,tws_n]^{\mathrm{T}}$。

3.2 由式(10)调整矩阵 $D_{n\times h}$。

4. 根据式(7)及矩阵 $D_{n\times h}$ 计算各句子间的相似度，并得到矩阵 $SD_{n\times n}$。

5. 根据 $SD_{n\times n}$ 构建基本的 TextRank 网络图 G。

6. 特殊段落的处理和特殊句子权重的传递，更新 $SD_{n\times n}$。

7. 更新图 G，并得到新的 TextRank 图 G'，在 G' 上运行 iTextRank 算法获得节点的收敛值矩阵 B_i。

8. 由式(11)得到矩阵 $FP_{n\times 1}$。

9. 依次根据 $TD_{n\times 1}$ 和 $FP_{n\times 1}$ 调整矩阵 B_i。

10. 输出句子重要度排序结果 P^*，并根据句子的先后顺序生成最终的摘要 D^*。

4.3 时间复杂度分析

下面分析 iTextRank 算法的时间复杂度。

(1)文本特征词预处理

给定一段中文文本，假设共有 n 个句子，全文共有 h 个特征词(以下所有计算步骤均以 n 和 h 为例)，即最终得到特征词词频矩阵 $D_{n\times h}$。这个过程主要的时间复杂度为 $T_1(n)=O(n\times h)$。

(2)初步计算相似度矩阵 $SD_{n\times n}$

需要计算句子两两之间的相似度，主要循环为：

```
for(int i=0;i<n;i++)
    for(int j=i+1;j<n;j++)
        calculate(Pi,Pj);
```

其中，calculate()函数利用余弦函数计算两个句子相似度，时间复杂度为 O(1)。这个过程总的时间复杂度为 $T_2(n)=O(n^2)$。

(3)特殊段落、关键句子权重的传递

将标记的特殊段落、关键句子进行权重传递，更新相似度矩阵的值。假设这些句子的数量为 $k(k<n)$，则时间复杂度为 $T_3(n)=O(k\times n)$。

(4)迭代计算，获得节点的收敛值向量 B_i

假设迭代次数为 t，并根据式(7)计算每个句子节点的权重，总的时间复杂度为 $T_4(n)=O(t\times n^2)$。

(5)调整最终节点权重矩阵 B_i

主要包括以下 3 种情况：

1)与标题有相似度的句子，即矩阵相乘 $B_i\cdot TD_{n\times 1}$；

2)特殊段落中的句子位置，即 $B_i\cdot FP_{n\times 1}$；

3)关键句子的处理。

由于每一步的时间复杂度均为常量级别，因此总时间复杂度为 $T_5(n)=O(1)$。

综上所述，总时间复杂度为 $T=T_1(n)+T_2(n)+T_3(n)+T_4(n)+T_5(n)=O(n\times h)+O(n^2)+O(k\times n)+O(t\times n^2)+O(1)=O(n^2)$。

显然，与经典的 TextRank 算法相比较，iTextRank 的时间复杂度并没有明显的提升。

5 实验

由于目前在中文摘要提取方面缺乏权威的评测样本，为了验证 iTextRank 的效果，本节设计了两个实验。实验一以百度百家上包含摘要的文章为样本，通过平均准确率、平均召回率和 F 值对比分析 iTextRank 与 TextRank 的效果。实验二将 iTextRank 与某在线摘要提取系统进行对比。

5.1 实验一：与 TextRank 的比较

首先通过网络爬虫从百度百家网站中抓取了若干篇文章，其中每篇文章都已包含文章作者自行提取的摘要。实验的目的是分别利用 TextRank 和 iTextRank 算法对文章的摘要进行提取，然后结合文章原有的摘要对比和分析它们的效

果。为了方便比较，实验过程中对文章的标题、摘要和正文进行了预处理，剔除一些摘要内容过短或摘要内容不是从原文中提取的文章。以句号为窗口结束的标志，每篇文章可划分为若干个句子，其中最短的文章有 15 句，最长的文章有 226 句，平均长度为 46.5 句，摘要的长度从 1 到 4 个句子不等。最终共取 3600 篇文章作为本实验的语料。具体统计信息如图 4 所示。

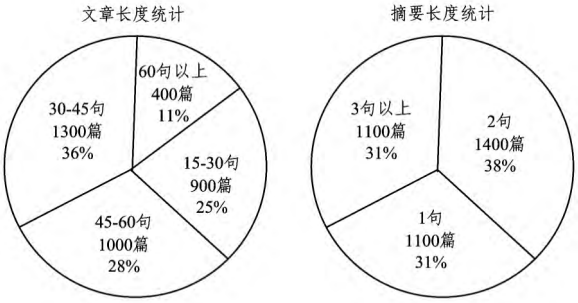

图 4　样本数据中文章及摘要长度统计图

将两种算法提取的摘要与原文章摘要进行比较，并计算平均准确率、平均召回率和平均 F 值。

(1)平均准确率 $P=\dfrac{\sum_{i=1}^{N}\dfrac{|x_i\cap y_i|}{|x_i|}}{N}$；

(2)召回率 $R=\dfrac{\sum_{i=1}^{N}\dfrac{|x_i\cap y_i|}{|y_i|}}{N}$；

(3)平均 F 值 $=\dfrac{P\times R\times 2}{P+R}$。

其中，x_i 表示用算法提取出来的第 i 篇的句子集，y_i 表示第 i 篇文章自带摘要形成的句子集，N 为文章总数，这里 $N=3600$。

iTextRank 中部分主要参数的值设置的如表 3 所列。

表 3　iTextRank 算法中的部分参数设置

参数名称	参数值
阻尼系数 d	d=0.85
阈值 e_1 和 e_2	$e_1=0.5$，$e_2=0.1$
特殊句子系数 a	a=1.1

实验结果如表 4 所列(其中每一列 a/b 分别对应着 iTextRank/TextRank 的值)。

表 4　iTextRank 和 TextRank 的实验结果

句数	平均准确率	平均召回率	平均 F 值
1 句	0.306/0.111	0.229/0.051	0.262/0.070
2 句	0.194/0.125	0.259/0.157	0.222/0.139
3 句	0.167/0.093	0.259/0.164	0.203/0.119
4 句	0.132/0.083	0.282/0.188	0.180/0.115
5 句	0.161/0.083	0.456/0.202	0.238/0.118

分析表 4 结果可知，iTextRank 算法比 TextRank 算法具有更高的准确率和 F 值，以及更低的召回率，这意味着 iTextRank 继承了 TextRank 无监督学习不需要依赖于外部的知识库或特定的语料库的优点，在不显著增加计算量的同时，抽取效果要好于后者，能够更加接近人工选取的结果。另外，通过人工观察可知，原文章中自带摘要均直接来自于文章中的第一段，且以第一句为主，而 iTextRank 摘要的句子分布较为均匀，具有较好的覆盖率，而且同时也说明 iTextRank 算法考虑特殊段落中句子的位置等信息是合理的。

5.2　实验二：与某在线自动摘要系统的比较

本实验主要是将 iTextRank 算法与某在线自动摘要系统(智呈科技：http://as.zhichengtech.com)进行比较。

5.2.1　对例 1 的分析

以前面例 1 中的 PageRank 介绍为测试样本，利用该在线摘要系统和 iTextRank 分别提取相同句数的摘要，具体结果如表 5 所列。

表 5　例 1 的摘要提取结果

句数	在线摘要系统	iTextRank
1	PageRank 并不计算直接链接的数量，它是将从网页 A 指向网页 B 的链接解释为由网页 A 对网页 B 所投的一票。	这样，PageRank 会根据网页 B 所收到的投票数量来评估该网页的重要性。
2	PageRank 并不计算直接链接的数量，它是将从网页 A 指向网页 B 的链接解释为由网页 A 对网页 B 所投的一票。此外，PageRank 还会评估每个投票网页的重要性，因为某些重要网页的投票被认为具有较高的价值，这样，它所链接的网页就能获得较高的价值。	PageRank 并不计算直接链接的数量，它是将从网页 A 指向网页 B 的链接解释为由网页 A 对网页 B 所投的一票。这样，PageRank 会根据网页 B 所收到的投票数量来评估该网页的重要性。
3	PageRank 并不计算直接链接的数量，它是将从网页 A 指向网页 B 的链接解释为由网页 A 对网页 B 所投的一票。此外，PageRank 还会评估每个投票网页的重要性，因为某些重要网页的投票被认为具有较高的价值，这样，它所链接的网页就能获得较高的价值。他们先假定所有网页的排名是相同的，并且根据这个初始值，算出各个网页的第一次迭代的排名，然后再根据第一次迭代排名算出第二次的排名。	PageRank 并不计算直接链接的数量，它是将从网页 A 指向网页 B 的链接解释为由网页 A 对网页 B 所投的一票。这样，PageRank 会根据网页 B 所收到的投票数量来评估该网页的重要性。但是问题又来了，计算其他网页 PageRank 的值需要用到网页本身的 PageRank 值，而其他网页的 PageRank 值反过来又影响本网页的 PageRank 的值，这不就成了一个先有鸡还是先有蛋的问题了吗？

从表 5 的结果来看，两者所提取出来的摘要句子均能较好地描述文章的主题思想和中心内容。由于只有一个测试样本，当只取一句话时由于两者计算的部分依据不同，因此所挑选的句子不同。当取两句话及以上时，均存在部分重叠的句子，这意味着两者均能较好地识别文本中的关键句子。而且，从结果来看，iTextRank 提取出来的摘要句子间的相关性更好一些，而且具有更好的概括性和可理解性。

5.2.2　基于新闻语料库的比较

首先随机地在百度新闻、新浪新闻、人民网、环球时报摘取部分文章以及标题作为本实验的语料，然后对比分析该在线摘要系统和 iTextRank 的提取效果。

为了便于计算，这里共取 3600 篇文章，其文章长度及占比的统计情况如图 5 所示。

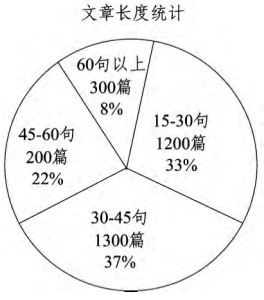

图 5　文章长度及占比统计

利用该在线摘要系统对样本数据 $D=\{D_i\}(1\leqslant i\leqslant 3600)$

中的每一篇文章 D_i 进行处理，均选取 3 个句子作为文章摘要，并记为 $oAbstract_i$。接着利用 iTextRank 算法提取出 D_i 指定句子数的摘要，记为 $iAbstract_i$。比较两者重叠的部分，分成以下 3 种情况：

① $iAbstract_i \bigcap oAbstract_i \geqslant 1$；

② $iAbstract_i \bigcap oAbstract_i \geqslant 2$；

③ $iAbstract_i \bigcap oAbstract_i \geqslant 3$，即 $oAbstract_i \subseteq iAbstract_i$。

为了便于比较，定义覆盖率：$T=m/N$，其中 m 为 $iAbstract_i$ 中符合某种情况的文章数，N 为总文章数。在本实验中，$N=3600$。

本实验中所有文章的平均句子数约为 43.53，我们认为排名在前 15 名的句子具有较好的代表性，因此在实验中取 15 为阈值来计算覆盖率。实验结果如表 6 所列。

表 6　3 种情况下的覆盖率统计(%)

句数	至少包含 1 句	至少包含 2 句	3 句以上全覆盖
1 句	33.33	0.00	0.00
2 句	63.89	11.11	0.00
3 句	72.22	25.00	2.78
4 句	77.78	41.67	2.78
5 句	80.56	52.78	8.33
6 句	83.33	63.89	11.11
7 句	86.11	69.44	13.89
8 句	94.44	72.22	19.44
9 句	94.44	75.00	22.22
10 句	97.22	86.11	27.78
11 句	97.22	91.67	44.44
12 句	97.22	91.67	58.33
13 句	97.22	91.67	63.89
14 句	97.22	91.67	69.44
15 句	97.22	91.67	72.22
15 句以上	100.00	100.00	100.00

对应的曲线图如图 6 所示。

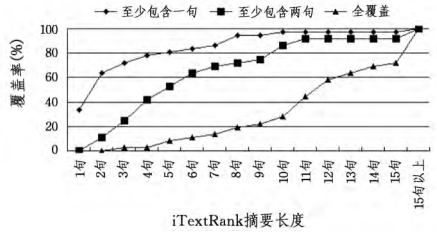

图 6　3 种情况下的覆盖率

从表 6 和图 6 可以看出，假设在线自动摘要系统的摘要句子数为 1，若 iTextRank 的摘要也只取 1 句，则覆盖率为 33.33%；若取 2 句，则覆盖率为 63.89%，之后随句子数的覆盖率增加不断地提高。这意味着由 iTextRank 算法和该在线自动摘要系统得到的中心句子具有较高的吻合度。另一方面，3 种情况下得到的覆盖率随着 iTextRank 选取句子的增加呈平稳上升的趋势，说明 iTextRank 算法具有较好的稳定性。

结束语　自动摘要是自然语言理解及文本挖掘中的一个研究重点和研究热点。本文在经典 TextRank 算法的基础上，将标题、段落、特殊句子、句子位置和长度等信息引入到 TextRank 网络图的构造中，即突出传统基于词频统计特征的优势，又结合有监督学习方法中的上下文及语义信息，本方法具有计算速度快、适应性强、易于实现和适合于大规模的文本摘要提取等优点。

下一步的工作将尝试把上述研究扩展到关键词提取及多文档自动摘要提取中，并结合主题思想提高摘要提取的准确率。

参 考 文 献

[1] Wang Ji-cheng, Wu Gang-shan, Zhou Yuan-yuan, et al. Research on Automatic Summarization of Web Document Guided by Discourse[J]. Journal of Computer Research and Development, 2003, 40(3): 398-405 (in Chinese)
王继成，武港山，周源远，等. 一种篇单结构指导的中文 Web 文档自动摘要方法[J]. 计算机研究与发展，2003，40(3)：398-405

[2] Zhang Qi, Huang Xuan-jing, Wu Li-de. A New Method for Calculating Similarity Between Sentences and Application on Automatic Text Summarization[J]. Journal of Chinese Information Processing, 2005, 19(2): 93-99 (in Chinese)
张奇，黄萱菁，吴立德. 一种新的句子相似度度量及其在文本自动摘要中的应用[J]. 中文信息学报，2005，19(2)：93-99

[3] Ji Wen-qian, Li Zhou-jun, Chao Wen-han, et al. Automatic Abstracting System Based on Improved LexRank Algorithm[J]. Computer Science, 2010, 37(5): 151-154 (in Chinese)
纪文倩，李舟君，巢文涵，等. 一种基于 LexRank 算法的改进的自动文摘系统[J]. 计算机科学，2010，37(5)：151-154

[4] Luo Wen-jun, Ma Hui-fang, He Qing, et al. Leveraging Entropy and Relevance for Document Summarization[J]. Journal of Chinese Information Processing, 2011, 25(5): 9-16 (in Chinese)
罗文娟，马慧芳，何清，等. 权衡熵和相关度的自动摘要技术研究[J]. 中文信息学报，2011，25(5)：9-16

[5] Li Ran, Zhang Hua-ping, Zhao Yan-ping, et al. Automatic Text Summarization Research Based on Topic Model and Information Entropy[J]. Computer Science, 2014, 41(11A): 298-300, 332 (in Chinese)
李然，张华平，赵燕平，等. 基于主题模型与信息熵的中文文档自动摘要技术研究[J]. 计算机科学，2014，41(11A)：298-300，332

[6] Mihalcea R, Tarau P. TextRank: Bringing Order Into Texts, 2004[C] // Proceedings of EMNLP 2004. Barcelona: ACM, 2004: 404-411

[7] Blanco R, Lioma C. Random Walk Term Weighting for Information Retrieval[C] // Proc. of the 30th SIGIR. New York: ACM Press, 2007: 829-830

[8] Blanco R, Lioma C. Graph-based Term Weighting for Information Retrieval[J]. Information Retrieval, 2012, 15(2): 54-92

[9] Lu Wei, Chen Qi-kai. An Information Retrieval Model Based on Weighted Graph and Sentence[J]. Journal of the China Society for Scientific and Technical Information, 2013, 32(8): 797-804 (in Chinese)
陆伟，程齐凯. 一种基于加权网络和句子窗口方案的信息检索模型[J]. 情报学报，2013，32(8)：797-804

[10] Wan X, Yang J, Xiao J. Towards an Iterative Reinforcement Approach for Simultaneous Document Summarization and Keyword Extraction[C] // Proc. of the 45th Annual Meeting of the Association of Computational Linguistics. Czech Republic: Association for Computational Linguistics, 2007: 552-559

[11] Yang Jie, Ji Duo, Cai Dong-feng, et al. Keyword Extraction in Multi-Document Based on TextRank Technology[C]// Proc. of NCIRCS'2008(The 2nd Vol.). 2008 (in Chinese)
杨洁,季铎,蔡东风,等. 基于 TextRank 的多文档关键词抽取技术[C]// 第四届全国信息检索与内容安全学术会议论文集(上). 2008

[12] Li Peng, Wang Bin, Shi Zhi-wei, et al. Tag-TextRank: A Webpage Keyword Extraction Method Based on Tags[J]. Journal of Computer Research and Development, 2012, 49(11): 2344-2351 (in Chinese)
李鹏,王斌,石志伟,等. Tag-TextRank:一种基于 Tag 的网页关键词抽取方法[J]. 计算机研究与发展, 2012, 49(11): 2344-2351

[13] Luhn H P. The Automatic Creation of Literature Abstracts [J]. IBM Journal of Research and Development, 1958, 2(8): 159-165

[14] Baxendale P E. Machine-made Index for Technical Literature-an Experiment[J]. IBM Journal of Research and Development, 1958, 2(4): 354-361

[15] Salton G, Wong A, Yan C S. A Vector Space Model for Automatic Indexing[J]. Communication of the ACM, 1995(18): 613-620

(上接第 193 页)

由图 10 可知在查询较优的已有算法中,MCL 算法的查询效率比网络划分算法高 15.26%,在节点数量较低情况 MCL 算法几乎与 SPB 算法相当,在节点数量增多时 MCL 算法比 SPB 算法效率高 15.65%。由此可见,MCL 算法适用于大规模的路网,尤其能在节点数量多的网络突显优势。图 11 以 W-US 数据集进行研究,显示的是在空间路网中节点的边数与查询效率之间的关系。

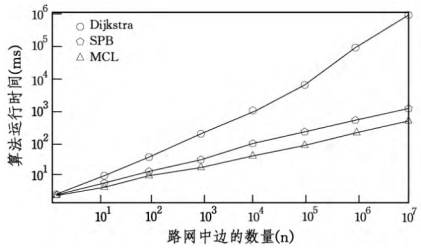

图 11 路网中的边数与查询效率的关系

从图中可以发现边的数量并不影响 MCL 算法的效率。

结束语 本文提出了一种基于虚拟路径和 MCL 格思想的查询算法,结合 ECT 树,达到了快速查询的目的,是查询效率高,所花的存储空间少。利用该算法能够实现快速查询的目的,特别是在大规模的城市路网中查询的高效性尤为显著。在此之前学者们所提出的算法的复杂度都是依赖城市路网节点的规模,而本算法的时间复杂度只与两点之间虚拟路径所经过的 MCL 格有关,因此省去了很多不需要扫描的节点,从而提升了算法的效率。下一步的工作就是如何在 KNN 查询中应用本算法。

参 考 文 献

[1] Dijkstra E W. A note on two Problems in connectionwithgraphs [J]. Numerische Mathematik, 1959, 1(1): 269-271

[2] Samet H, Sankaranarayanan J, Alborzi H. Sealable network distance browsing in spatial databases [C]// The 8th SIGMOD ACM Conference. 2008: 43-54

[3] Klein P N, Mozes S, Weimann O. Shortest Paths in direeted Planar graphs with negative lengths: Aline-space time algorithm [J]. ACM Transactions on Algorithms, 2010, 6(2): 1-13

[4] Papadias D, Zhang J, Mamoulis N, et al. Query processing in spatial network databases [C]// The 3rd VLDB. 2003: 802-813

[5] Cho H J, Chung C W. An efficient and scalable approach to CNN queries in a road network[C]// The 5th VLDB. 2005: 865-876

[6] Sanders P, Sehultes D. Highway hierarchieshasten exact shortest path queries[C]// 13th Annual European Symposium2005. Palma de Mallorca, Spain, October 2005: 568-579

[7] Geisberger R, Sanders P, Sehultes D, et al. Con-Traction hierarchies[C]// Faster and Simpler Hierarchical Routing in Road Networks. WEA, 2008: 319-333

[8] Bast H, Funke S, Matijevie D. Transit: ultrafast shortest-Path queries with linear-time Preprocessing[C]// The 9th DIMACS Implementation Challenge. 2006: 175-192

[9] Deng Ding-xiong. Shortest Path Queries on Road Networks based on Pre-Computation Techniques [D]. Shanghai: FuDan University, 2011

[10] Lee K C K, Lee W C, Zheng Bai-hua, et al. ROAD: A New Spatial Object Search Framework for Road Networks [J]. IEEE Transactions Onknowledge and Data Engineering, 2012, 24(3): 547-560

[11] Wang Shi-ming. Research and Realization of the Shortest Path Algorithm within Topical Urban Road Network[D]. Shandong: Shandong University, 2012

[12] Lu Zhao, Shi Jun. Design and Implementation of parallel shortest path search algorithm[J]. Computer Engineering and applications, 2010, 46(3): 69-71(in Chinese)
卢照,师军. 并行最短路径搜所算法的设计与实现[J]. 计算机工程与应用, 2010, 46(3): 69-71

[13] Deng Ding-xiong. Shortest Path Query for Road Network Based on SPB Tree[J]. Computer Engineering, 2011, 37(22): 56-63

[14] Gargantini I. An Effective Way to Represent Quadtrees [J]. Communication of ACM, 1982, 25(12): 905-910

[15] Sankaranarayanan J, Alborzi H, Samet H. Efficient query Processing on spatial networks[C]// GIS'05. 2005: 200-209

[16] Sankaranarayanan J, Samet H. Distance oracles for spatial network[J]. IEEE Computer Society, 2009, 9: 652-663

[17] Sankaranarayanan J, Samet H, Alborzi H. Path oracles for Spatial networks[J]. PVLDB, 2009, 2: 1210-1221

[18] Goldberg A V, Harrelson C. Computing the shortest Path: A * search meets graph theory [C]// Proceedings of the Annual ACM-SIAM Symposium on Discrete Algorithms. 2005: 156-165