SOM 算法、LVQ 算法及其变体综述*)

Survey on SOM Algorithm、LVQ Algorithm and their Variants

张敏灵　陈兆乾　周志华

(南京大学计算机软件新技术国家重点实验室　南京210093)

Abstract Self-Organizing Mapping is an important neural network model, it has been applied widely in areas such as pattern recognition, signal processing, data mining, etc. LVQ algorithm is a kind of supervised clustering method which originates from SOM. In recent years, many variants of SOM and LVQ have been proposed. In this paper, SOM, LVQ, and some of their variants are surveyed.

Keywords Neural networks, SOM algorithm, LVQ algorithm

1. 引言

SOM[1](Self-Organizing feature Map,简称SOM)网络是一种自组织竞争型人工神经网络,它是由著名神经网络专家T. Kohonen 教授于1981年提出的,因此 SOM 网络又称为 Kohonen Map。

SOM 算法是一种非监督(unsupervised)的聚类方法,自20年前该算法提出至今,很多研究者围绕该算法在模式识别,信号处理,数据挖掘等理论和应用领域做了大量工作,并且取得了大量研究成果[1]。这些成果的取得很大程度上归功于 SOM 算法本身的简明性和实用性。

本文将首先描述 SOM 算法及其变体。LVQ[2](Learning Vector Quantization)算法的基本思想源于 SOM 算法,作为 SOM 算法的一种重要扩展,本文也对 LVQ 算法及其几种变体进行介绍。

2. SOM 算法及其变体

SOM 网络的典型拓扑结构如图1所示,它由输入层和输出层(竞争层)两层网络组成(本文以典型的二维输出层为例),两层神经元之间实现全互连,输入层有 N 个神经元,输出层有 M 个神经元,并均匀地排列成矩形。

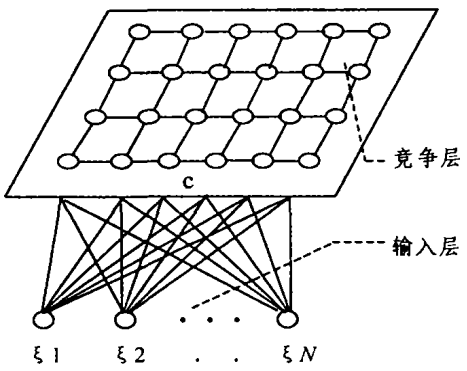

图1　SOM 网络的典型拓扑结构

记所有输出神经元 c 组成的集合为 Φ,神经元 c 与输入层神经元之间的连接权向量为 W_c。SOM 算法作为一种非监督聚类方法,理论上可以将任意维的输入模式在输出层映射成一维、二维甚至更高维的离散图形,并保持其拓扑结构不变。

该算法的聚类功能主要是通过以下两个简单的规则实现的[2]:

1)对于提供给网络的任一个输入向量 ξ,确定相应的输出层获胜神经元 s,其中 $s=\arg\min_c \|\xi-W_c\| \forall c\in\Phi$。

2)确定获胜神经元 s 的一个邻域范围 N_s,按如下公式调整 N_s 范围内神经元的权向量:$W_c=W_c+\varepsilon(\xi-W_c)\forall c\in N_s$,该调整过程使得 N_s 内神经元的权向量朝着输入向量 ξ 的方向靠拢。

随着学习的不断进行,学习率 ε 将不断减小,邻域 N_s 也将不断缩小,所有权向量将在输入向量空间相互分离,各自代表输入空间的一类模式,这就是 Kohonen 网络特征自动识别的聚类功能。

自 SOM 算法提出至今,出现了多种该算法的变体,下面就其中几种常见的变体做一个介绍。

2.1 Growing SOM

由前面的介绍可知,SOM 算法作为一种通用的聚类方法,理论上可以完成任意输入空间向量的特征自动提取。但是将该算法应用于特定的数据集时却存在两个问题[3]:一是在训练开始之前就要求确定一个合适的网络大小是很困难的。二是预先确定好的输出层拓扑结构(通常情况下是矩形)可能无法很好地反应输入数据的特征。针对这两个问题,人们提出了一种新的 SOM 网络—Growing SOM[3](Growing Self-Organizing network)。该算法的基本思想是把 n 维空间的单形体,比如一维空间的线段,二维空间的三角形,三维空间的四面体作为基本的构建模块,随着训练的不断进行,渐进、动态地生成 SOM 网络。

图2就是将 Growing SOM 算法应用于一个圆形概率分布的数据集上所得到的最初几步结果[5]。

由上例可以看出,随着学习过程的不断进行,SOM 网络也在不断"增长",平均每隔200个训练例就有新的单形体(上例为三角形)插入到 SOM 网络中,那么这些单形体应该在什么位置插入就成了 Growing SOM 算法所面临的一个关键问题,一般而言,可以通过下述方法解决[3]:

·对于整个 SOM 网络定义一个可以度量的期望目标。

·对于每一个输出神经元(比如上例中的三角形的顶点),

*)本文工作得到江苏省自然科学基金(BK2001406)资助。

估计它们对网络期望目标贡献的大小。

·在那些对期望目标贡献不大的神经元附近插入新的神经元。

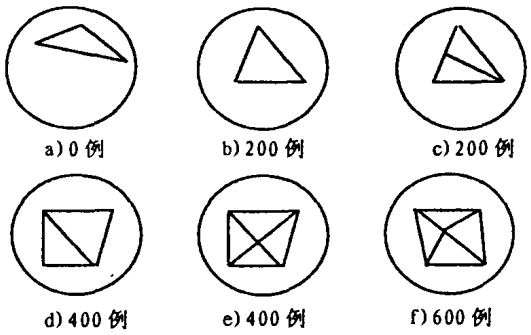

图2 Growing SOM 应用于圆形概率分布的数据

对于资源有限的 SOM 网络(比如限定了最大的网络规模),就必须引入相应的神经元删除准则,限于篇幅,这里就不介绍了。

总的来说,Growing SOM 与普通的 SOM 网络的不同点在于:网络结构是动态生成的;持续不变的自适应能力;固定的邻域范围。相同点在于:固定的网络维数;邻域的确定取决于网络的拓扑结构。

2.2 TS-SOM

TS-SOM[7](Tree Structured Self- Organizing Maps)是 SOM 算法的一种快速实现。TS-SOM 的网络结构可以用一个图 $T=(V,E1,E2)$来表示,其中 V 是结点(神经元)的集合,$E1$是用于"纵向"层次连接的边构成的集合,而 $E2$是用于各层之间"横向"连接的边构成的集合。对于每一个节点 $i\in V$,都存在一个相应的子树 $T_i=(V_i,E1_i,E2_i)$,并且满足如下条件:$V_i\subset V$,$E1_i\subset E1$,$E2_i\subset E2$,$T_i\cap T_j=\phi$(如果 $i\notin V_j$ 且 $j\notin V_i$)。

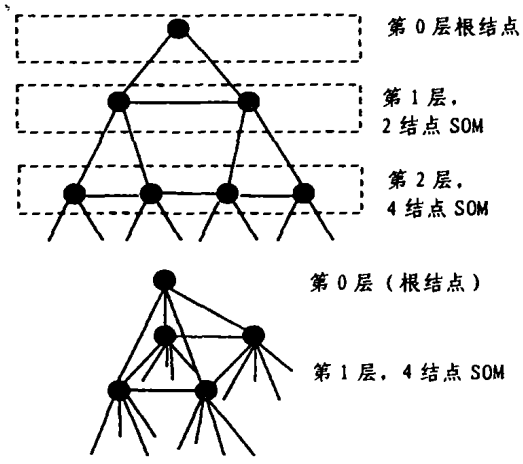

图3 一维及二维的 TS-SOM

图3是一维和二维的 TS-SOM 的拓扑结构。在 TS-SOM 中,树的每一层都是一个 SOM 网络,并且每一个结点都会引导出2^n个子结点,其中 n 为 TS-SOM 的维数,由此可知 TS-SOM 的第 l 层包含$(2^n)^l$个结点($l=0,1,2\cdots\cdots L-1$,L 为 TS-SOM 的层数)。

由于树结构本身的递归性,在对第 l 层上的神经元进行匹配运算时,可以利用前面 $l-1$层网络所提供的信息以减少匹配次数。J. H. Friedman[6]等人指出,对于总共含有 N 个结点的网络和任一输入模式 ξ,普通 SOM 网络需要 $O(N)$的时间以找出对应的获胜神经元,而 TS-SOM 只需 $O(\log_p N)$的时间就可以完成匹配,其中 p 是 TS-SOM 中每个神经元子结点的数目,且 $p=2^n$。因此,TS-SOM 实际上是将树结构有效地应用于 SOM 算法当中,给出了 SOM 算法的一种快速实现。

2.3 G-SOM

在达尔文进化论和门德尔-摩根遗传学说的启示下,美国 Michigan 大学的 J. Holland[11]于70年代提出了遗传算法,其基本思想是:首先以字符串对问题的解进行编码,然后在选择、杂交和变异等遗传算子作用下通过对字符串群体的进化获取对现实世界问题的较优解。

G-SOM[9](SOM together with genetic algorithm)将遗传算法应用于 SOM 网络优化,通过对初始 SOM 群体不断进化,最终能够得到一个较优解。在应用遗传算法时,有3点值得注意[9]:

1)染色体的编、解码。对于任一个 SOM 网络,它所对应的染色体中应该包含该网络的权值信息,并体现出网络的拓扑结构,有时 SOM 网络训练规则中所用的学习参数也可编码存入染色体中。

2)适应度函数及个体适应值的确定。对于一个前馈型神经网络,可以采用网络的模拟输出和实际输出之间的均方误差值作为其个体适应值。但是对于 SOM 网络而言,由于它是一种非监督的聚类算法,因此要找出一种合适的适应度函数是比较困难的,一般来说可以从下面两点来考虑[10]:①适应度函数应该能够反映出个体 SOM 网络在训练过程中所经历的自组织步骤的多少;②适应度函数应该能够度量个体 SOM 网络实现拓扑映射的效果。针对上述两点有许多种适应度函数可供选择,限于篇幅,这里就不做介绍了。

3)个体网络的训练。对于个体的 SOM 网络,可以根据标准的训练参数进行训练,也可以根据染色体中提供的训练参数进行训练,这完全取决于是否将网络的训练参数编码至染色体中。

实验表明[9],通过将遗传算法应用于 SOM 的优化,最终得到的 SOM 网络具有较优的初始权向量及较好的学习参数,具有很好的网络性能。

3. LVQ 算法及其变体

LVQ 算法是对 SOM 算法的一种扩展,它的基本思想源于 SOM 算法,它对应的网络结构与 SOM 很相似,但并不像 SOM 网络那样存在某种特定的拓扑结构。LVQ 算法是一种监督型的聚类方法,该算法与 SOM 算法最大的区别在于提供给 LVQ 网络的每个训练例都有一个"标记"(label),该"标记"用于指明每个训练例所属的类别,在网络的训练过程中起到一定的监督作用。因此,LVQ 算法实际上是 SOM 算法基本思想在监督学习邻域中的一种应用。

LVQ 算法对应的网络结构如下:

·包含 n 个输入神经元,其输入向量为 $X=(x_1,x_2,\cdots,x_n)$,X 所对应的类别记为 T。

·每个输出神经元 j 都对应一个权向量 $W_j=(w_{1j},w_{2j},\cdots,w_{nj})$,记所有输出神经元 j 构成的集合为 Ω。

·C_j 为输出神经元 j 所代表的类别,不同的输出神经元可以代表同一个类别。

LVQ 网络期望通过对带有"标记"的训练例的学习,能够

正确预测提供给网络的测试例的分类。LVQ 网络的训练过程如下：

1)初始化权向量 $W_j \forall\ j \in \Omega$，设置初始学习率 $\alpha = \alpha(0)$

2)从训练集中选取一输入向量 X，找出与 X 具有最小 Euclidean 距离的权向量 W_k，其中 $k = \mathrm{argmin}_j \|X - W_j\| \forall\ j \in \Omega$

3)按如下规则调整神经元 k 的权向量：

·如果 $T = C_k$(即输出神经元 k 所代表的类别与输入模式一致)$W_k(new) = W_k(old) + \alpha(X - X_k(old))$

·如果 $T \neq C_k$(即输出神经元 k 所代表的类别与输入模式不一致)$W_k(new) = W_k(old) - \alpha(X - W_k(old))$

4)训练集中选取另外一个输入向量提供给 LVQ 网络，返回步骤2)直到所有的向量都提供了一遍为止。

5)减小学习率 α，并且测试停止条件是否满足，如果满足则停止训练，否则返回步骤2)。

由过程3)中所采用的权向量调整策略可以看出，当被选中的输出神经元 k 对应的类别和输入向量 X 所对应的类别一致时，将调整权向量使其向输入向量的方向靠拢，反之，调整权向量使其偏离输入向量。过程5)中所指的停止条件一般为训练达到了固定的迭代次数或者学习率降至预设的最小值。

上面介绍了什么是 LVQ 算法，它作为与 SOM 对应的监督型聚类算法，也存在多种变体，下面对其中几种常见的变体做一个介绍：

3.1 LVQ2

在本文第三部分介绍的 LVQ 算法中，对于某个输入向量 X，算法只对与其具有最小 Euclidean 距离的权向量 W_k 进行调整。

在 LVQ2[2]中，算法还要考虑与 X 具有次近 Euclidean 距离的权向量 W_r，X 与 W_k 及 W_r 间的距离分别记为 d_k 与 d_r。如果下述3个条件都满足的话，算法就对 W_k 及 W_r 同时进行调整，否则就按照原先的 LVQ 算法调整权向量 W_k：

1)W_k 及 W_r 代表不同的分类。

2)次近权向量 W_r 与 X 代表同一个分类。

3)d_k 与 d_r 大致相等。一般来说，如果 d_k 与 d_r 能够满足 $d_k/d_r > (1-\varepsilon)$且 $d_r/d_k < (1+\varepsilon)$，我们就认为 d_k 与 d_r 是大致相等的，其中 ε 的取值依赖于训练例的多少。

在上述3个条件都满足的情况下，按下述公式调整 W_k 及 W_r，使得 W_k 远离输入向量 X，而 W_r 向输入向量的方向靠近：

1)$W_k(new) = W_k(old) - \alpha(X - W_k(old))$

2)$W_r(new) = W_r(old) + \alpha(X - W_r(old))$

LVQ2算法通过同时考察两个权值向量 W_k 及 W_r，可以加快算法的收敛速度，使得各个权向量快速的向目标位置移动。

3.2 LVQ2.1

LVQ2.1[2]在 LVQ2的基础上做了一些改进，LVQ2.1也是同时考察与某个输入向量 X 最近的两个权向量 W_{c1} 及 W_{c2}，但并不关心 W_{c1} 及 W_{c2} 哪一个离 X 更近(对应的距离分别为 d_{c1} 及 d_{c2})。当下面两个条件同时满足时将对 W_{c1} 及 W_{c2} 进行调整：

1)W_{c1} 与 W_{c2} 中，有一个权向量代表的分类和 X 所表示的一致，而另一个不一致。

2)$\max[d_{c1}/d_{c2}, d_{c2}/d_{c1}] < (1+\varepsilon)$且 $\min[d_{c1}/d_{c2}, d_{c2}/d_{c1}] > (1-\varepsilon)$。

如果上述条件满足，不妨设 W_{c1} 与 X 代表的类别相同，算法将调整权向量 W_{c1} 及 W_{c2}，使得 W_{c1} 向输入向量 X 的方向靠近，而 W_{c2} 远离输入向量，调整公式为：

1)$W_{c1}(new) = W_{c1}(old) + \alpha(X - W_{c1}(old))$

2)$W_{c2}(\mathrm{new}) = W_{c2}(old) - \alpha(X - W_{c2}(old))$

3.3 LVQ3

LVQ3[2]也是同时考虑与输入向量 X 距离最近的两个权向量 W_{c1} 及 W_{c2}。当条件 $\min[d_{c1}/d_{c2}, d_{c2}/d_{c1}] > (1-\varepsilon)(1+\varepsilon)$ 满足时(ε 的一个典型取值为0.2)。权向量将按照下述规则进行调整：

·如果 W_{c1} 及 W_{c2} 两个权向量中有一个对应的分类与输入向量 X 一致，另一个不一致，则权向量的调整规则同 LVQ2.1。

·如果 W_{c1} 与 W_{c2} 代表相同的分类，则权向量 W_{c1} 及 W_{c2} 均采用公式 $W_c(new) = W_c(old) + \beta(X - W_c(old))$ 进行调整，其中 $\beta = m\alpha$，$0.1 < m < 0.5$。

LVQ3算法通过对标准 LVQ 算法学习过程的修改，可以使得在网络学习过程不断进行的同时，网络的权向量能够很好地反映输入空间的概率密度分布，并且防止权向量偏离最优的位置。

3.4 G-LVQ

G-LVQ[2]将遗传算法应用于 LVQ 算法中，以克服 LVQ 算法本身的一些缺陷。

优化一个 LVQ 网络可以从优化它的初始权向量，学习算法以及网络规模(权向量的数目)3个方面来考虑，总的来说，LVQ 网络本身的学习算法已经很理想了，所以只从优化初始权向量及网络规模两方面来考虑。为了将遗传算法用于 LVQ 网络的优化，必须对个体 LVQ 网络编码，编码时，个体 LVQ 网络对应的染色体中应包含网络的权向量及其对应的分类信息；在得到初始的染色体群体以后，即可应用遗传算法进行进化，经过若干代之后找出具有最高适应值的个体作为最优解。

在 G-LVQ 中，有两个问题是值得注意的：

1)个体适应值的确定。在 G-LVQ 中，个体 LVQ 网络的适应值的确定应该综合考虑多方面的因素，包括该网络的分类正确率，权向量的数目，以及网络的失真等等。一般而言，网络的分类正确率是在确定适应值时优先考虑的因素。

2)遗传算法的改进。为了更好地应用遗传算法，除了应用常规的变异及杂交算子外，G-LVQ 又添加了三种遗传算子：

·复写算子。对于个体 LVQ 网络而言，如果它的某个权向量在训练过程中"胜出"的次数明显大于其它权向量，则可应用复写算子，先将该权向量进行变异，然后再把变异体进行编码添加到相应的染色体中。

·灭除算子。对于个体 LVQ 网络，如果它的某个权向量在训练的过程中"胜出"的次数少于其它任何一个权向量，或者根本无法"胜出"，则可利用灭除算子，将该权向量的编码信息从对应的染色体中删除。

·随机增长算子。在遗传算法执行过程中，可以应用随机增长算子，随机的增加某个 LVQ 网络的权向量(对应的分类也是随机产生的)，将随机生成的权向量的信息编码放入对应的染色体中。

将遗传算法和 LVQ 算法结合起来可以有效地避免普通神经网络学习中存在的局部最小问题，并且可以更快更精确地得到特定问题的解。

结束语 本文首先介绍了 SOM 算法以及它的几种变体，紧接着对 SOM 算法的一种重要扩展－LVQ 算法进行了

描述，最后对 LVQ 算法的几种变体也做了介绍。

除了已经介绍的几种以外，还有许多由 SOM 算法发展而来的变体，例如 ASSOM[13]（Adaptive-Subspace SOM），PSOM[14]（Parametrized SOM），SSOM[15]（Stochastic SOM），GTM[16]（Generative topographic mapping）等；最近发展起来了一种新兴的 SOM 变体－WEBSOM[17]，主要用于对 Internet 上存在的海量文档进行组织，在所有已经实现的应用系统中，最多的已经对大约700万份文档作了有效的组织。限于篇幅，对于这些变体就不一一介绍了。

参 考 文 献

1 Kohonen T. Self-organizing maps. 2nd edition. Berlin：Springer，1997

2 Kohonen T. Self-organizing and associative memory. Heidelberg：Springer，1984

3 Fritzke B. Growing self-organizing networks-history，status quo，and perspectives. In：Oja E.，Kaski S. eds. Kohonen maps，Amsterdam：Elservier，1999. 131～144

4 Fritzke B. Growing cell structure- a self-organizing network for unsupervised and supervised learing. Neural Networks，1994，7(9)：1441～1460

5 Fritzke B. Let it grow-self-organizing feature maps with problem dependent structure. In：Kohonen T.，Makisara K.，Simula O.，Kangas J. eds. Artificial Neural Networks，Amsterdam：Elservier，1991. 403～408

6 Friedman J H，Bentley J L，Finkel R. A. An algorithm for finding best matches in logrithmic time. ACM Trans. Math.，1977，Software 3：209～216

7 Koikkalainen P. Tree structured self-organizing maps. In：Oja E.，Kaski S. eds. Kohonen maps，Amsterdam：Elservier，1999. 121～130

8 Koikkalainen P，Oja E. Self-organizing hierarchical feature maps. In：Proc. of the Intl. Joint Conf. on Neural Networks，San Diego，1990. 279～284

9 Polani D. On the optimization of self-organizing maps by genetic algorithm. In：Oja E.，Kaski S. eds. Kohonen maps，Amsterdam：Elservier，1999. 157～169

10 Polani D. Organization measures for self-organizing maps. In：Proc. of the Workshop on Self-Organizing Maps，Helsinki，1997. 280～285

11 Holland J H. Adaption in natural and artificial systems. Ann Arbor：The University of Michigan Press，1975

12 Ritter H. Self-organizing maps on non-euclidean spaces. In：Oja E.，Kaski S. eds. Kohonen maps，Amsterdam：Elservier，1999. 97～109

13 Kohonen T，Kaski S，Lappalainen H. Self-organized formation of various invariant- feature filters in the adaptive- subspace som. Neural Computation，1997，9(6)：1321～1344

14 Walter J，Ritter H. Rapid learning with parametrized self-organizing maps. Neurocomputing，1996，12(2-3)：131～153

15 Graepel T.，Obermayer K. A stochastic self-organizing map for proximity data. Neural Computation，1999，11(1)：139～155

16 Bishop M.，Svenson M.，Williams K. I. Gtm：the generative topographic mapping. Neural Computation，1998，10(1)：215～234

17 Kaski S，Honkela T，Lagus K，Kohonen T. Websom-self-organizing maps of document collection. Neurocomputing，1998，21(1-3)：101～117

18 Kaski S，Kangas J，Kohonen T. Bibliography of self-organizing (som)papers：1981-1997. Neural Computing Surveys，1998，1(3-4)：1～176

19 Thiran P. Kohonen self-organizing map with quantized weights. In：Oja E.，Kaski S. eds. Kohonen maps，Amsterdam：Elservier，1999，145～156.

20 胡守仁，余少波，戴葵. 神经网络导论. 长沙：国防科技大学出版社，1993

21 张立明. 人工神经网络的模型及其应用. 上海：复旦大学出版社，1993

（上接第112页）

直接转换成存储地址了。

理论分析和试验结果证明：P 应取小于存储区容量的素数（质数）。例如对前所述的四个关键字，若存储区为000到999，则 P 应取为小于1000的素数，取 P＝997，则可得以下结果：

关键字	内部代码	H(k)＝k MOD 997
KEyA	11052501	756
KEyB	11052502	757
AKEy	01110525	864

可见，这些结果是比较好的。所以，除留余数法是经常使用的。

数字分析法构造哈希函数的思想是：对各个关键字内部代码的各个码位进行分析。若第 i 位上，各个关键字中出现的数码种类比较多，则可选中该值为哈希函数值的一部分。需要选多少位来组成哈希函数值，应视存储区地址范围而定。这种方法比较简单、直观，但需要预先知道关键字的情况，这就限制了它的应用范围。

解决冲突的方法有两大类：一类称为开放地址法，当发生冲突时，用某种方法形成一个探测的序列，沿着这个序列一个一个单元地查询，直到找到这个关键字或找到一个开放的地址（即没有进行存储的空单元），如果是插入操作，则遇到空单元就可进行插入操作；如果是查询操作，遇到空单元就是查询失败。另一类称为链地址法，当发生冲突时，就拉出一条链，建立一个链接方式的子表，则具有相同哈希函数值的关键字被链接在一个子表中。

2. 禁用词表的使用

禁用词表是很小的，没有价值作为索引词的词构成表。一般地，介词、冠词、连词、副词和代词等虚词都在所收 i 列，据统计，大约有120个左右。如：of，if，able，about，after，a，been，Certainly 等。禁用词表按字母顺序排列，并计算出每个词的长度，将禁用词表存入计算机中，作为一个待查用的禁用词表文件，对于禁用词表，要有建表和修改表的程序进行控制。

3. 抽取词干

在建立索引过程中，抽取词干是很重要的，它可在索引过程中减少词数，即可减少存储空间。抽取词干主要采取移去后缀（有时是前缀）。例如从一个文档里来的词：reformation、reformative、reformatory、refored 和 reformism，抽取词干得到字根：reform，所有这5个词在索引里将映射到 reform。在索引过程中节省了4个词的存储空间。

已经发明了几种自动抽取词干的算法，例如：除去词缀（最长匹配、简单除去），后继变化，表查找和 n-gram。

参 考 文 献

1 Berry M W，Browne M. Understanding Search Engines Mathematical Modeling and Text Retrieval . Philladephla ：Society for industrical and Applied Mathematics(www . siam . org)，1999

2 储荷婷，张晓林，王芳. Internet 网络信息检索－原理工具技巧. 北京：清华大学出版社，1999. 10

3 William B Frakes Ricardo Baeza-Yates . Information Retrieval . USA：Prentice Hall PTR，1992