[19] 中华人民共和国国家知识产权局

[51] Int. Cl[7]
G06F 17/27
G06F 17/30

[12] 发明专利申请公开说明书

[21] 申请号 200510059231.0

[43] 公开日 2005 年 9 月 21 日

[11] 公开号 CN 1670723A

[22] 申请日 2005.3.16
[21] 申请号 200510059231.0
[30] 优先权
[32] 2004. 3.16 [33] US [31] 10/801,968
[71] 申请人 微软公司
地址 美国华盛顿州
[72] 发明人 E·D·布里尔 S·-P·库塞赞

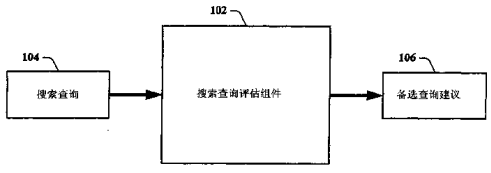
[74] 专利代理机构 上海专利商标事务所有限公司
代理人 沈昭坤

权利要求书 5 页 说明书 24 页 附图 10 页

[54] 发明名称 改进的拼写检查系统和方法

[57] 摘要

本发明利用搜索查询字符串的迭代变换以及从搜索查询日志和/或 web 数据提取出来的统计量，给搜索查询字符串提供可能的备选拼写。这提供一种能用来给每个用户提供个性化建议的拼写检查方法。通过利用搜索查询日志，本发明能考虑到辞典里没有但仍可接受为所关心的搜索查询的子串。这允许一种提供在辞典内容之外的质量更高的备选拼写建议的方法。本发明的一个实例通过利用从查询日志中提取出来的词一元模型和/或二元模型的统计量并结合迭代搜索，在子串级别上工作。这为给定查询提供了实质上比只利用子串匹配的方法更好的拼写备选方案。本发明的其他实例能接收来自除了搜索查询输入之外的来源的输入数据。



ISSN1008-4274

1. 一种促进拼写检查的系统，包括：

一接收包含文本的输入数据的组件；以及

一拼写检查组件，其识别文本中一组潜在地被错拼的子串，并基于至少一个查询日志，给所述子串组建议至少一个备选拼写；所述查询日志包括在一个时间范围内被诸用户用于查询数据集的数据。

2. 如权利要求 1 所述的系统，其特征在于，所述拼写检查组件在建议至少一个备选拼写时进一步利用依赖于用户的信息。

3. 如权利要求 1 所述的系统，其特征在于，所述子串组的备选拼写进一步基于至少一个可信辞典；所述可信辞典包括从由一个有内容的可信辞典和一个没有内容的可信辞典组成的组中选择的至少一个。

4. 如权利要求 3 所述的系统，其特征在于，所述拼写检查组件进一步使用一个停用词列表；所述停用词列表包括从由一个有内容的停用词列表和一个没有内容的停用词列表组成的组中选择的至少一个。

5. 如权利要求 4 所述的系统，其特征在于，所述有内容的停用词列表包括一个包含高频词和功能词及其常见拼写错误的停用词列表。

6. 如权利要求 4 所述的系统，其特征在于，所述拼写检查组件使用迭代过程来搜索备选拼写空间。

7. 如权利要求 6 所述的系统，其特征在于，所述拼写检查组件至少部分地使用试探法来将限制强加于用来确定建议的备选拼写的搜索空间。

8. 如权利要求 7 所述的系统，其特征在于，所述试探法至少部分地利用至少一个边缘来限制所述搜索空间。

9. 如权利要求 4 所述的系统，其特征在于，所述查询日志包括在一个时间范围内被要求的查询的一个直方图。

10. 如权利要求 9 所述的系统，其特征在于，所述被要求的查询的直方图涉及诸用户的一个子集；所述子集包括至少一个用户。

11. 如权利要求 9 所述的系统，其特征在于，所述查询日志驻留在服务器计算机上。

12. 如权利要求 9 所述的系统，其特征在于，所述查询日志驻留在客户

计算机上。

13. 如权利要求 9 所述的系统，其特征在于，所述拼写检查组件利用来自至少一个查询日志的子串出现和共现统计数据。

14. 如权利要求 13 所述的系统，其中子串包括从由在至少一个可信辞典之内的条目、在一停用词列表中的条目以及没有一组预定义的定界符的字符序列组成的组中选择的至少一个。

15. 如权利要求 13 所述的系统，其特征在于，所述子串共现统计数据包括子串二元模型计数；子串二元模型包括文本中的一对子串。

16. 如权利要求 15 所述的系统，其特征在于，所述子串二元模型包括文本中的一对毗邻子串。

17. 如权利要求 16 所述的系统，其特征在于，所述有内容的停用词列表的子串共现统计数据进一步包括一个带停用词序列忽略计数的子串二元模型。

18. 如权利要求 13 所述的系统，其特征在于，所述来自查询日志的子串出现和共现统计数据被存储于一相同的可搜索数据结构中。

19. 如权利要求 18 所述的系统，其特征在于，所述数据结构包括一个特里结构。

20. 如权利要求 18 所述的系统，其特征在于，所述系统以与其处理个别子串时所用的相同方式处理串接的和/或拆分的子串。

21. 如权利要求 20 所述的系统，其特征在于，所述拼写检查组件产生一组备选拼写，所述一组备选拼写是从由至少一个查询日志和至少一个辞典所组成的组中选择的至少一个的子串。

22. 如权利要求 21 所述的系统，其特征在于，所述备选拼写组包括通过迭代改正过程确定的一组备选拼写。

23. 如权利要求 22 所述的系统，其特征在于，所述迭代改正过程包括多个将至少一个子串转变成另一个子串作为备选拼写的迭代；当所有可能的备选拼写都不如当前的备选拼写组恰当时，所述迭代改正过程停止。

24. 如权利要求 23 所述的系统，所述备选拼写及其适合程度基于一概率串距离和一统计上下文模型计算。

25. 如权利要求 24 所述的系统，其特征在于，所述概率串距离包括一个修正的上下文依赖的加权 Damerau-Levenshtein 编辑函数，当指点改变时，

所述编辑函数允许字符的插入、删除、替换、毗邻调换和远距离移动。

26. 如权利要求 24 所述的系统，在每个迭代中，用于一子串的所述备选拼写组通过利用从至少一个查询日志和至少一个可信辞典提取出来的可搜索子串数据结构来产生。

27. 如权利要求 26 所述的系统，在每个迭代中，用于每个子串的所述备选拼写组被限制在距离输入子串的概率距离 δ 内；所述限制被强加在每个迭代内，而没有对迭代改正过程作为整体进行限制。

28. 如权利要求 27 所述的系统，在每个迭代中，所述迭代改正过程通过利用统计上下文模型来搜索最佳备选拼写组。

29. 如权利要求 28 所述的系统，其特征在于，所述统计上下文模型包括从至少一个查询日志提取出来的子串出现和共现统计数据。

30. 如权利要求 29 所述的系统，其特征在于，在每个迭代中，使用 Viterbi 搜索来促进根据所述上下文模型确定所述最佳备选拼写组。

31. 如权利要求 30 所述的系统，其特征在于，在一次迭代中，所述 Viterbi 搜索可以使用边缘来限制备选拼写搜索，使得对每对毗邻子串，如果其中任何一个子串在至少一个可信辞典之内，则在所述迭代中只允许改变所述子串中的一个。

32. 一种促进拼写检查的方法，包括：

接收包含文本的输入数据；

识别文本中的一组潜在地被错拼的子串；以及

基于至少一个查询日志，为所述子串组建议至少一个备选拼写；所述查询日志包括在一个时间范围内被诸用户用于查询数据收集的数据。

33. 如权利要求 32 所述的方法，其特征在于，所述子串组的所述备选拼写进一步基于至少一个可信辞典；所述可信辞典包括从由一个有内容的可信辞典和一个没有内容的可信辞典组成的组中选择的至少一个。

34. 如权利要求 33 所述的方法，进一步包括：

至少部分地使用一个停用词列表来促进确定至少一个备选拼写；所述停用词列表包括从由一个有内容的停用词列表和一个没有内容的停用词列表组成的组中选择的至少一个。

利用来自至少一个查询日志的子串出现和共现统计数据；所述查询日志包括在一个时间范围内被要求的查询的一个直方图，来自所述查询日志的子

串出现和共现统计数据存储在一个相同的可搜索数据结构中。

以与处理单个子串所用的相同方式来处理串接和/或拆分的子串；以及

产生一组备选拼写，所述备选拼写组是从由至少一个查询日志和至少一个辞典所组成的组中选择的至少一个中的子串，所述备选拼写组包括通过迭代改正过程确定的一组备选拼写。

35. 如权利要求 34 所述的方法，其特征在于，所述迭代改正过程包括：

将至少一个子串转变成另一个子串，作为备选拼写；以及

当所有可能的备选拼写都不如当前的备选拼写组恰当时，停止所述迭代改正过程；基于随机串距离和统计上下文模型，计算所述备选拼写及其适合程度。

36. 如权利要求 35 所述的方法，在所述迭代改正过程的每个迭代中进一步包括：

利用从至少一个查询日志和至少一个可信辞典之内提取出来的可搜索子串数据结构，来产生用于一个子串的所述备选拼写组；

将每个子串的所述备选拼写组限制在距离一输入子串的随机距离 δ 内；所述限制被强加在每个迭代内，而没有将迭代改正过程作为整体进行限制；以及

通过利用统计上下文模型搜索一组最佳备选拼写；所述统计上下文模型包括从至少一个查询日志提取出来的子串出现和共现统计数据。

37. 如权利要求 36 所述的方法，进一步包括：

在每次迭代中，使用 Viterbi 搜索来促进根据所述上下文模型确定所述最佳备选拼写组；在一次迭代中，所述 Viterbi 搜索可以使用边缘来限制备选拼写搜索，使得对于每对毗邻子串，如果其中任何子串在至少一个可信辞典之内，则所述迭代中只允许改变所述子串中的一个。

38. 一种便于对搜索引擎的查询进行拼写检查的系统，包括：

接收包含文本的输入数据的方法；以及

识别文本中一组潜在地被错拼的子串并基于至少一个查询日志给所述子串组建议至少一个备选拼写的方法；所述查询日志包括一个时间范围内被诸用户用于查询数据收集的数据。

39. 在两个或多个计算机组件之间传输的促进搜索查询拼写检查的数据包；所述数据包至少部分地包含与搜索查询拼写检查系统有关的信息，所述

拼写检查系统至少部分基于至少一个查询日志，给一组查询字符串至少部分地提供至少一个备选拼写。

40. 一种计算机可读介质，所述介质中存储有如权利要求 1 所述的系统的计算机可运行组件。

41. 一种使用如权利要求 32 所述的方法的设备，其特征在于，所述设备包括从由计算机、服务器和手持式电子设备组成的组中选择的至少一个。

42. 一种使用如权利要求 1 所述的系统的设备，其特征在于，所述设备包括从由计算机、服务器和手持式电子设备组成的组中选择的至少一个。

改进的拼写检查系统和方法

(1)技术领域

本发明一般地涉及拼写检查，尤其涉及通过利用查询日志来改良拼写检查的系统和方法。

(2)背景技术

与自动化程序、系统和服务的交互，已经成为大多数人生活中日常事务的一部分——尤其随着因特网的出现。例如，对于某些人来说，网络冲浪或浏览可能是“新的”全国性的娱乐。依照这样的系统，像文字处理那样的应用程序已经帮助许多人在他们各自的工作中或者在他们的个人生活中变得更加有效率，例如给朋友键入一封信或电子邮件。这些应用程序已经增加了许多自动化的功能组件，例如用于实质上以具有任何所需要的字体、色彩、形状或表格的格式化文档的工具。已经被许多用户所了解并普遍接受的一种工具是拼写检查应用程序，用户从文字处理软件中调用拼写检查应用程序来检查各个文档的全部或部分，和/或当用户打字时调用拼写检查应用程序，使其在后台运行，检查拼写。通常，为了执行正确的拼写检查，拼写检查应用程序可以使用一个“有效字符串”的字典。如果拼写检查程序遇到一个不在字典之内的字符串，就假定该字符串是拼写错误，并试图在字典中为该错拼的字符串找到“最接近”的字符串。大多数拼写检查程序给用户提供一个可能匹配的列表，由此，如果匹配在该列表中，用户可以从列表中选择具有改正过的拼写的词。拼写检查的其他功能组件可以执行自动改正——如果用户这样配置的话。

然而，文字处理的拼写检查只展现了潜在领域的部分情景，该潜在领域可以用于协助用户把信息输入到文件或文档内。例如，对所有可用的潜在网络站点和服务，用户经常通过明确地键入站点名称的全部或部分，或通过执行对出现在网页标题或主体上的字或短语的搜索，在站点之间导航。正如许多人已经认识到的那样，如果站点信息或搜索查询输入不正确，再次导航的时间代价就会变得相当高。搜索引擎或其他应用程序中使用的语言处理程序经常处理用户

的查询，并尝试区分真实的用户命令和不正确输入的信息。然而应该明白，被输入到搜索引擎中用于查询的信息类型可能在结构或形式上与文字处理应用程序中通常使用的信息类型有很大不同。因此，文字处理程序中用来在有些单独和孤立的基础之上检查词的工具在用于从一般的查询数据中生成信息时，可能很少甚至没有效用。

浏览器或其他信息搜索查询给拼写检查应用程序提出了一个独特的问题，因为查询常常包含在标准的拼写检查字典中找不到的词，如艺术家、产品或公司名称等。另外一个问题是查询中的词可能已经被不正确地输入，但是没有错误拼写（例如，“and processors”而不是“amd processors”）。因此，人们在像搜索引擎的输入框那样的键入行中输入文本的方式，常常与文字处理中的键入有很大不同。查询输入中所输入的内容和人们所犯的错误类型也自然有很大不同。此外，网络数据和搜索查询在性质上是非常动态的，包含大量的固有名词；新的产品、人、机构、地点和事件每天都变得流行。同样地，标准字典尽管适用于文字处理中上下文的拼写检查，却可能并不适用于键入行和搜索—查询中的拼写检查。

字典（即辞典）是任何拼写检查程序的重要组件，因为其中包含的信息提供了确定不正确拼写的基础。然而，对于许多需要进行拼写检查的应用程序（如提供给输入框的文本输入）来说，标准字典对这个问题不是最佳的。例如，为了对输入到搜索引擎的输入框中的文本进行拼写检查，字典应该包括诸如“hanging chad（悬空票）”和“Apolo Anton Ohno（阿波罗·安东·奥诺）”之类的字符串，以便检查近来的事件或可能所关注的信息。可以知道，这些字符串和大量其他类型的字符串不会在标准字典中出现。一种可能的途径是在用户输入到诸如搜索引擎或语言处理程序之类的特定位置的内容的日志上使用子串匹配技术。不幸的是，这种途径的一个问题是查询日志通常也将会包含大量的输入错误，并返回与用户所需搜索无关的子串匹配。

另外，拼写检查所用的字典和搜索的上下文总是在改变。这些动态的行为不能通过利用传统字典和搜索查询处理来解决。例如，如果现在有一个名为Limp Bizkit的流行乐队，对“bizkit pictures（bizkit照片）”的查询很可能是指的是这个乐队，而不是“biscuit（饼干）”的错误拼写。如果这个乐队突然变

得不流行，而且有一本关于饼干照片的畅销书，那么“bizkit pictures”就更可能是“biscuit pictures”的错误拼写。同样，在当前的政治状态下，如果 arnold 现在是受欢迎的加利福尼亚州州长，“governor anld”可能指的是“govenor arnold（州长 Arnold）”。因此，搜索查询的上下文显著地影响着拼写检查。

(3)发明内容

下面给出本发明简化了的内容，以便提供对本发明一些方面的基本理解。该内容不是本发明的广泛纵览。该内容并不是要鉴别本发明的关键/重要的元素或描绘本发明的范围。其唯一目的是以简化的形式给出本发明的一些概念，作为后面给出的更加详细的描述的前奏。

本发明一般地涉及拼写检查器，尤其涉及通过利用查询日志来改进拼写检查的系统和方法。搜索查询字符串的迭代变换连同从搜索记录和/或 web 数据提取出来的统计量一起，被用来给搜索查询字符串提供可能的备选拼写。这提供一种能用来给每个用户提供个性化建议的优异的拼写检查方法。通过利用搜索查询日志，本发明能解释辞典里没有但仍可接受为所关心的搜索查询的子串。这允许一种提供在辞典内容之外的质量更高的备选拼写建议的方法。本发明的一个实例通过利用从查询日志中提取出来的词一元模型和/或二元模型的统计量以及迭代搜索，在子串水平上工作。这为给定查询提供了实质上比只利用准确子串匹配的方法更好的拼写备选方案。因此本发明，例如，能基于流行的概念/查询的最近历史，调整它所建议的备选方案。本发明也能基于相关的先前查询日志，为给定用户调整其改正，提供更加恰当的拼写备选方案。本发明的其他实例能接收来自除搜索查询输入之外的来源的输入数据。这提供了一种利用查询日志促进在普通文字处理器等等的上下文中进行拼写检查的方法。

为实现上述相关成果，结合下列说明和附图，在此描述本发明的某些示例性方面。然而，这些方面只指示可能应用本发明原理的各种方法中的一些，本发明规定为包括所有这些方面及其等效。结合附图一起考虑，从下面的本发明具体实施方式可以明显看出本发明的其他优点和新颖特征。

(4)附图说明

图 1 是一个依照本发明一个方面的搜索查询评估系统的方框图。

图 2 是另一个依照本发明一个方面的搜索查询评估系统的方框图。

图 3 是一个依照本发明一个方面的搜索过程的示例。

图 4 是另一个依照本发明一个方面的搜索过程的示例。

图 5 是一个依照本发明一个方面的信息流结构的示例。

图 6 是一个依照本发明一个方面的促进搜索查询的一种方法的流程图。

图 7 是另一个依照本发明一个方面的促进搜索查询的一种方法的流程图。

图 8 是又一个依照本发明一个方面的促进搜索查询的一种方法的流程图。

图 9 例示了一个本发明可以在其中运行的操作环境的实例。

图 10 例示了另一个本发明可以在其中运行的操作环境的实例。

(5)具体实施方式

现在参考附图描述本发明，其中始终用相同的参考数字来指示相同的元素。在下列描述中，出于解释的目的，阐明了很多特定的细节，以提供对本发明的彻底理解。然而，显然本发明可能不需要这些特定的细节就能实现。在其他实例中，以方框图的形式把众所周知的结构和器件显示出来，以方便描述本发明。

本说明书中，术语“组件”意指与计算机有关的实体，可以是硬件、硬件和软件的组合、软件或执行中的软件。例如，一个组件可以是但不限于是，一个在处理器上运行的进程、一个处理器、一个对象、一个可执行程序、执行的一个线程、一个程序和/或一个计算机。作为示例，运行在服务器上的应用程序和服务器本身都可以是计算机组件。一个或多个组件可以驻留在一个进程和/或执行的一个线程中，并且，组件可以位于一个计算机内和/或分布在两个或更多的计算机之间。“线程”是在操作系统内核调度执行的进程内的实体。正如本领域中众所周知的那样，每个线程有相关的“上下文”，即与线程的执行相关的易失性数据。线程的上下文包括系统寄存器的内容和属于该线程所处进程的虚拟地址。因此，包含线程上下文的真实数据随着线程执行而发生改变。

本发明给对搜索引擎的查询进行拼写检查提供改进的系统和方法。本发明的一个实例使用一个可信辞典（该辞典为一种语言的一个有效词列表）和查询日志作为数据源。本发明通过利用从查询日志提取出来的词一元模型和二元模型（忽略停用词）统计量以及一种新型的迭代查询，为给定查询提供比只用字符串匹配时更好的拼写备选方案。本发明的另一个实例至少部分地利用低计数词 n 元模型（n 元模型定义为 n 个连续词的序列；特别地，一元模型为单独的

词，二元模型是两个连续词的序列）的 web 统计量。

传统的词拼写改正依赖于文本书写所用的该语言的可信辞典 L 和距离函数 d。给文本中每个不在辞典之内的词形提出在辞典内的一组备选拼写 $\{w_1, w_2, ..., w_k\}$ 的建议，这些有效的备选方案出现在给定的固定距离阈值 δ 之内，通常相差为一个或两个编辑（即是说，$dist(w, w_i) \leq \delta$）。备选方案通常按条件似然函数 $P(w_i | w)$ 排序，该条件似然函数考虑到该语言中词的概率（通常由目标语言的大型语料库的最大似然估计计算得出）和词之间的距离。

实际的距离函数 d 和阈值 δ 对拼写检查程序的准确度很重要。一个极端是，使用限制太严格的函数/阈值的组合将导致无法为给定查询找到最好的改正。另一个极端是，使用限制较少的函数可能得到很不可靠的改正建议。本发明提供一种可行的折中办法，通过在字符串水平上利用对备选方案的修正的有限搜索，但不在实质上限制在词水平上搜索备选方案。本发明的一个实例利用上下文依赖的加权 Levenshtein 距离，它允许字母的插入、删除、替换、毗邻调换和长距离移动，作为基本编辑。阈值本身是变量，依赖于每个词的特征（主要是，词是否在辞典之内）和迭代数目。

图 1 中，显示了一个依照本发明一个方面的搜索查询评估系统 100 的方框图。搜索查询评估系统 100 包括搜索查询评估组件 102。搜索查询评估组件 102 接受搜索查询输入 104 并输出备选查询建议数据 106。搜索查询评估组件 102 利用辞典和搜索查询日志来评估每个搜索查询。本发明的其他实例也利用 web 统计量来评估搜索查询输入 104。通常利用迭代过程来进一步精选每个备选建议，直到获得最佳建议。搜索查询日志提供统计信息，而搜索查询评估组件 102 可以利用该统计信息，给典型的用户或给定的用户查找最佳解决方案。在本发明的又一个实例中，对搜索查询日志进行处理，以便于可以使用定时模式来促进搜索查询的评估。定时模式可以是但不限于是年模式、日期模式（例如，每个星期三、假日等等）和日模式等等的时间。因此，本发明的一个实例能获得本年度十月份和上一年度十月份的查询日志。这种时间模式识别的类型有利于产生较好的搜索查询建议数据 106。同样地，本发明能利用其他类型的模式识别，例如，用户的搜索模式、用户的兴趣和嗜好、用户的首选项等等。本发明可以获得与给定用户的这些特定方面有关的搜索查询日志和/或网络数据，用于进一步促进搜索查询评估。

在本领域中的技术人员将会明白，除了搜索查询输入之外，本发明的范围

还包括来自其他来源导出的输入数据。因此，本发明的实例可以用于接收来自文字处理程序、电子邮件程序、即时消息程序和聊天室程序等等的输入数据。以这样的方式，除了搜索查询程序之外，查询日志还可以用于其他环境中。因此，当在本发明的上下文中使用时，术语“搜索查询”的各种形式和“输入数据”的各种形式是同义的。因此，输入数据是指提交给本发明的实例、用于拼写检查目的的任何数据。

参见图 2，图示了另一个依照本发明一个方面的搜索查询评估系统的方框图。该搜索查询评估系统包括接收搜索查询数据 204 和输出备选查询建议数据 206 的搜索查询评估组件 202。搜索查询评估组件 202 由子串处理组件 208 和迭代查询评估组件 210 组成。子串处理组件 208 接收搜索查询数据 204，并将其标记化（tokenize）到可供迭代查询评估组件 210 使用的水平。迭代查询评估组件 210 利用查询日志数据 214，也能利用诸如辞典数据 212 和/或可任选的 web 统计量数据 216 那样的其他数据源。该数据被用于提供搜索查询数据 204 的增强评估，将在下面进一步详细说明。迭代查询评估组件 210 处理标记化的查询词，并把一组新的备选搜索查询子串 218 发回给子串处理组件 208 以供标记化。这一迭代过程继续进行，直到迭代查询评估组件 210 确定已经达成最佳建议，并将这一信息输出为备选查询建议数据 206。

为更好地认识本发明，理解其使用的上下文及其目的是有帮助的。用户发送给 web 搜索引擎的查询大约有 10%由于拼写错误而失效。因此，web 搜索的一个重要问题是检测和改正拼错的查询。对搜索引擎的查询进行拼写检查是一个与传统的文档拼写检查（例如，典型的文字处理的拼写检查程序）根本不同的问题。这个过程的一些重要特性的把它和传统文档拼写改正区分开来：

- 不像能给每个错误拼写的词提供一组备选方案的传统词拼写，web 查询拼写检查程序只能给一个 web 查询建议一个拼写备选——这意味着需要比传统词拼写程度所提供的高得多的精度。
- 传统的拼写检查程序使用可信辞典并把注意集中到在辞典之外的词。在 web 查询中，未知词（根据这样的辞典）可能不代表拼写错误，它们对 web 搜索来说可能是有效的（例如，limp bizkit）。查询词列表不是一个固定目标，而是快速变化的。当然也存在这样的情况，即基

于搭配信息，在辞典之内的词应该改变成其他在辞典之内或甚至在辞典之外的词（例如，food explorer→ford explorer，limp biz kit→limp bizkit）。

- 在 web 搜索中，拼错的词可能较预期词相差有许多字母编辑，尤其是当这些词表示人、公司、技术或产品的名字时候，仅仅提到其中一些。
- 正如本发明所提供的那样，除了可信辞典和/或语料库之外，web 查询拼写改正还受益于查询日志的存在。该查询日志包含关于词频、词上下文和拼写错误的近乎实时的重要信息。在传统的拼写中没有利用这样的资源。

本发明将对 web 查询进行拼写改正的问题用公式表示为，根据从 web 查询日志和 web 数据提取出来的统计量，将查询字符串迭代变换成其他字符串，这些其他字符串越来越表示合适的查询。通用拼写改正的任务有很长的历史，传统上集中在解决印刷错误，诸如字母的插入、删除、替换和调换（如，McIlroy, M.D.; Development of a Spelling List; *J-IEEE-TRANS-COMM*, 30(1); 91-99; 1982）。典型的拼写检查程序为每个未知词（即，在本语言的可信辞典里找不到的词）计算出一小组在辞典之内的备选方案，作为可能的改正建议。这样的系统通常忽略上下文，而只依赖于关于在辞典之内的词频（从一个大语料库中估计）和最平常的错误的信息，都在词的水平（例如，用 *acceptible* 而不是 *acceptable*）和字符的水平（例如，误用 *f* 而不是 *ph*）上。

本发明中网络查询拼写改正的任务在许多方面不同于传统的拼写检查，并因此带来了不同的挑战。查询的有效性不能由简单的辞典检索或检查其语法性来决定。大多数的 web 查询包含一个概念或多个概念的枚举，许多时候都包含合理但却在辞典之外的词。例如，web 查询的改正程序应该能够以关于别人的搜索内容的信息为基础，建议“*lego tos*”改正为“*lego toys*”这一最好的可能改正（有争议），尽管基于字母的错误模型（如典型的文字处理拼写检查程序中所用的）会为错误拼写 *tos* 预测 *toss*、*toes*、*tops*、*tons* 和 *togs* 是更可能的备选方案，且在许多英文辞典之内可能不会出现词“*lego*”。

仅仅定义什么是有效的 web 查询就代表一项相当困难的事业。很清楚，可信辞典不能单独使用，因为每天都有许多新的名字和概念变得流行，而且维持

高覆盖率的辞典即使不是不可能的，也是特别困难的。一个可行的替代方案可以是收集和开发数百万使用 web 和 web 搜索的人的经验。因此，词的有效性不是由辞典给定，而是由它出现在人们所查询的内容中的频率给定，类似于 Wittgenstein 的观察（见，Wittgenstein, L.; Philosophical Investigations; Macmillan, New York, third edition; 1968）所述，“一个词的意思取决于它在语言中的具体运用”。但是，这样的方式有其自己的警告。简单地从 web 查询日志提取出频率在特定阈值之上的所有那些查询信息就认为其有效，这将是错误的。例如，错误拼写 *britny spears* 是一个比拼写正确但可能包含在辞典之外的词（如查询 *bayesian nets* 或 *amd processor*）的更加常见的查询。同样要对于计算包含被查询词的 web 文档的数目也成立。非常常见的词语的拼错频率要远高于那些相当有效而又比较不常见的词语。

许多在网络查询中出现的错误表现为上下文不适当的有效词替换（例如，*principal* 和 *principle*），这不能够由辞典检索或基于单独词频来发现。Kukich 的观察（见，Kukich, K; Techniques for Automatically Correcting Words in Text; *ACM Computing Surveys*, 24(4):377-439; 1992）认为，在现代文档中发现的错误大约有 25-50%实际上是这类替换，可以预言，他的结论似乎也适用于查询日志数据，如果不是更高程度的话。在上下文依赖的拼写改正（CSSC，context-sensitive spelling correction）的框架内，解决这样的替换错误已经成为早先各种 NLP（Natural Language Processing，自然语言处理）工作[如，(Golding, A.R.; A Bayesian Hybrid Method for Context-Sensitive Spelling Correction; *Proceedings of the Workshop on Very Large Corpora*, pages 39-53; 1995); (Golding, A.R. 和 D. Roth; Applying Winnow to Context-Sensitive Spelling Correction; *Proceedings of the 13th International Conference on Machine Learning*, pages 182-190; 1996); (Mangu, L. 和 Brill, E.; Automatic Rule Acquisition for Spelling Correction; *Proceedings of the 14th International Conference on Machine Learning*, pages 734-741; 1997); 以及 (Cucerzan, S. 和 Yarowsky, D.; Augmented Mixture Models for Lexical Disambiguation; *Proceedings of EMNLP 2002*, pages 33-40; 2002)]的目标。虽然获得了有希望的结果（92-95%的准确度），但是早先工作的范围非常有限（最多 18 个混淆组，两个或三个备选方案，例如，{*peace*，*piece*}），而且所研究的算法严重依赖于注释数据的存在，利用从这些数据提取出来的大量上下文特征。在 web 查询拼写改正的情况下，没有注释数据，但是可以得到大量以查询日志（查询日志是在

一段时间内发送到搜索引擎的查询的直方图）形式存在的非注解数据。web 查询的另一个特征是非常短（平均 2.2 个词），难以应用利用大量基于相对较宽的上下文窗口的特征的技术。而且，在 CSSC 中所利用的典型数据通常被认为是没有其他上下文拼写错误和替换错误，这不是 web 查询的实际情况。

其他的内容和性能进一步限制了在网络查询拼写改正系统设计中使用计算强度密集算法。例如，在基于服务器的架构中，这样的系统必须满足限制非常严格的时间性能要求，而空间的要求可以放松。下面提供这些限制的一部分，但没有提供对这些限制的完全分析。通过利用任务的一系列正式定义并给出显示与每种情形相对应长处和局限的具体例子，简要分析“典型的”拼写改正方法的例子。该问题被迭代地重新定义，开始于纯粹地基于可信辞典的方法，结束于可信辞典的作用大大减小的方法。这样做的时候，为了提供有效网络查询的可用定义，给出具体的前进步骤。

令 Σ 为一种语言的字母表，$L \subset \Sigma^*$ 为该语言的一个广泛覆盖的辞典。于是，拼写改正最简单的定义可以如下：

给定 $w \in \Sigma^* \setminus L$，找到 $w' \in L$，使得 $dist(w,w') = \min_{v \in L} dist(w,v)$。

即是说，对于文本中任何在辞典之外的词，找到可用辞典之内最接近的词形，并假定其为正确的拼写备选方案。*dist* 可以是任何基于字符串的函数，例如，在两个词中非共有字母个数和共有字母个数之间的比率。在拼写改正中用得最多两类的距离是 Damerau（见， Damerau, F.J.; A Technique for Computer Detection and Correction of Spelling Errors; *Communications of ACM*, 7(3):171-176; 1964）和 Levenshtein（见， Levenshtein, V.I.; Binary Codes Capable of Correcting Deletions, Insertions and Reversals; *Doklady Akademii Nauk SSSR*; 163(4) p845-848; 1965）提出的编辑距离，以及相关矩阵距离（见， Cherkassky, V.; Vassilas, N.; Brodt, G.L.; Wagner, R.A.; 和 Fisher, M.J.; The String to String Correction Problem; *Journal of ACM*, 21(1):168-178; 1974）。本发明的一个实例使用 Damerau-Levenshtein 编辑距离的修正形式，如下文介绍所示。

上述公式表示并没有考虑一种语言中的词频。该问题的一个简单解决办法

是计算目标语言 L 中词的概率，作为对 L 上的大语料库 C 的最大似然估计（MLE）。通用的拼写改正问题可以再用公式表示如下：

给定 $w \in \Sigma^* \setminus L$，找到 $w' \in L$，使得 $dist(w,w') \le \delta \; P(w') = \max_{v \in L: dist(w,v) \le \delta} P(v)$ 。

在这一公式表示中，所有处于未知词某个“合理的”距离 δ 内的在辞典之内的词被认为是好的候选词，基于其先验概率 $P(\cdot)$ 选择它们中最好的那个。

一个自然的前进步骤是用后验概率作为噪声通道模型框架内的目标函数（见， Kernighan, M.; Church, K.; 和 Gale, W.; A Spelling Correction Program Based on a Noisy Channel Model; *Proceedings of COLING 1990*）。该目标函数是以原来的拼写为条件的改正的概率 $P(v|w)$，并包括了将一种语言中词的先验概率 $P(v)$ 和错误拼写词形 v 作为 w 的概率，$P(w|v)$。为简单起见，在此描述的所有公式表示均使用先验概率作为目标函数，但是在实践中，可以用后验概率来代替。

到目前为止所考虑的公式表示并不考虑上下文——在孤立中改正未知词。这是个问题，因为在非常多的情况中，上下文对拼写改正是极为重要的，如下面的例子所例示：

power crd → *power cord*

video crd → *video card*

根据其上下文，错误拼写的词形 *crd* 应该被改正为两个不同的词（其他有效备选方案如 *video cd* 等是存在的；由于它们对本讨论没有用，被有意忽略了）。传统的拼写检查程序（例如，典型的文字处理的拼写检查程序）所建议的改正在两种情况下都是同样的：*card*、*cord*、*crud*、*curd*、*cud*，以这样的顺序。当只允许有一个建议（这在 web 查询拼写改正中是非常典型的）时，总是从这个列表中采用第一个建议会导致许多错误。

考虑上下文问题的一个可能公式表示如下：

给定 $s\in\Sigma^*$， $s=c_l w c_r$，且有 $w\in\Sigma^*\setminus L$ 及 $c_l,c_r\in L^*$，找到 $w'\in L$，使得 $dist(w,w')\le\delta$ 且 $P(c_l w' c_r)=\max_{v\in L:dist(w,v)\le\delta} P(c_l v c_r)$。

为简单起见，在这一公式表示中空格和其他的词定界符被忽略了。随后问题的公式表示也将忽略词定界符，虽然可以认为词的标记化是拼写改正过程的一个重要部分。

基于以上定义的传统拼写—改正系统不处理用其他有效词对有效词进行替换的情况。在 web 搜索的情况中，当改正比原来的查询更“富有意义”时，给有效词提供改正建议很重要，例如：

food explorer → *ford explorer*

golf war → *gulf war*

下面的例子显示传统拼写改正程序不处理的两个问题的组合，即上下文敏感的改正和有效词替换：

chicken sop → *chicken soup*

sop opera → *soap opera*

CSSC 的任务部分地解决这些问题，如 NLP 文献中所定义的，可以用公式表示如下：

给定一种语言中的一组易混淆的有效词形 $W=\{w_1,w_2,...,w_n\}$ 和一个字符串 $s=c_l w_i c_r$，选择 $w_j\in W$，使得 $P(c_l w_j c_r)=\max_{k=1..n} P(c_l w_k c_r)$。

正如公式表示的那样，相对于传统的通用拼写改正。CSSC 的任务更倾向于涉及词义消歧（WSD，word sense disambiguation）。然而当所有的词 w' 满足 $dist(w,w')\leq\delta$ 时，通过为文本中的每个词构造一组易混淆词，可以将该任务应用于拼写改正。短语拼写改正的一般化问题可以用公式表示如下：

给定 $s\in\Sigma^*$，找到 $s'\in L^*$，使得 $dist(s,s')\leq\delta$ 且 $P(s')=\max_{t\in L^*:dist(s,t)\leq\delta} P(t)$ 。

通常，当 $s\notin L^*$ 时（即是说，组成的词中至少有一个是未知的），拼写改正是希望的，但如上所示，还有经常发生的情况如 *sop opera*，此时有效词形序列应该改变为有效词形的其他序列。可以观察到词的边界被隐藏在这一最近的公式表示中，将它变得更一般化并允许它覆盖 web 查询改正的其他重要拼写错误，也就是串接和拆分。例如：

power point slides → *powerpoint slides*

waltdisney → *walt disney*

chat inspanich → *chat in spanish*

这一公式表示仍然不覆盖一类重要的拼写改正，这类拼写改正必须由 web 查询拼写改正系统处理，并表现为实际上是给定上下文中的有效形式的未知词（因此，$s'\notin L^*$），例如：

amd processors → *amd processors*

典型的文字处理程序会加亮显示 *amd* 为错误拼写词，并提供改正建议，诸如：*mad*、*amid*、*am*、*and* 和 *ad*。在 web 查询拼写改正的情况下，系统不应该建议拼写改正，因为上面的短语代表一个合理的查询，尽管事实上它包含未知词。

最近的公式表示中没有处理的一些甚至更有趣的情况是有效词应该改变成未知词（即是说，不在可信辞典之内），如下面的例子中，两个有效词应该

串接成一个未知词：

gun dam planet → *gundam planet*

limp biz kit → *limp bizkit*

这就引出拼写—改正问题的更一般化的公式表示，如下所示：

给定 $s \in \Sigma^*$，找到 $s' \in \Sigma^*$，使得 $dist(s,s') \le \delta$ 且 $P(s') = \max_{t \in \Sigma^*: dist(s,t) \le \delta} P(t)$ 。

这一公式表示并不明确地使用该语言的辞典，虽然在字符串的似然估计 $P(s)$ 中仍然可能用到辞典。这意味着，在 web 查询改正的情况中，书写查询所用的实际语言变得没有比从中可以估计字符串概率的未经注释的查询日志训练数据重要。因此，这个概率模型可以是对像 web 查询那样的无意义字符串的量度的一种替代。这样，在任何传统语料库中似乎不合情理的随意的名词短语如 *sad tomatoes*（一个乐队的名字），在网络搜索的上下文中变得有意义了。

传统的词拼写—改正依赖于可信辞典和字符串距离函数。前面给出的拼写改正公式表示利用了这样的字符串距离和阈值来限制搜索备选拼写的空间。先前的各种工作已经解决选择适当的字符串距离函数的问题。本发明一个实例使用修正的上下文依赖的加权 Levenshtein 距离，当指点发生改变时加权 Levenshtein 距离允许字母的插入、删除、替换、毗邻调换和长距离移动。

实际的距离函数 d 和阈值 δ 对拼写程序的准确度很重要。使用限制太严格的函数/阈值的组合会导致不能为给定查询找到最好的改正。例如，使用标准的 Levenshtein 距离（定义为将一个字符串变换为另一个字符串所需要的指点改变次数的最小值，此处指点改变是下列操作中的一个：插入一个字母、删除一个字母以及用一个字母来替换另一个字母）并且阈值 $\delta = 1$，*donadl duck* → *donald duck* 的改正将是不可能的。然而，利用限制较少的函数可能得到非常不可靠的改正建议。例如，利用同样的经典 Levenshtein 距离和 $\delta = 2$ 将允许字符串 *donadl duck* 的改正，但是也将会导致错误的改正如 *log wood* → *dog food*（基于加入在

*P(s)*中的查询频率）。但是，在多样性的情况中，距离大的改正仍然是适当的，例如：

例 1： *platnuin rings* → *platinum rings*

例 2： *ditroitigers* → *detroit tigers*

在第一个例子中，典型的文字处理拼写检查程序可能只建议 *plantain* 和 *plantains* 是错拼词 *platnuin* 的改正。在第二个例子中，典型的文字处理拼写检查程序会加亮显示 *ditroitigers* 为错误拼写，但没有提供改正建议。尽管传统的可信辞典和语料库的方式可能无法解决这种类型的问题，但它可以由本发明通过利用大的查询日志来解决。

根据距离和选择阈值，如果像 *ditroitigers* 那样的拼写错误距离正确的备选方案太远，那么就不可能在一个步骤中就找到正确的备选方案。然而，使用本发明一个实例，通过允许中间的有效改正步骤，可以得到正确的备选方案，如 *ditroitigers* → *detroittigers* → *detroit tigers*。问题的最后一个公式表达并没有明确地利用该语言的辞典。然而，基于查询相对频率和备选拼写，训练所用的查询日志中出现的任何子串都可以被认为是有效的改正，并可以被建议为当前 web 查询的备选方案。事实上，与文字处理程序所用的典型拼写检查程序相对，本发明的基本拼写检查程序（本发明的基本拼写检查程序是一个非迭代系统）的一个实例会建议 *detroittigers*，因为这个备选方案频繁地出现在查询日志中。另一方面，如果作为独立的查询提供给基本检查程序，通过使用类似的作为本发明迭代改正方式的基础的查询日志频率事实，*detroittigers* 本身可以被改正为 *detroit tigers*。这种方式的本质是查询日志的三个典型特性：（1）查询日志中的词以各种方式被错误拼写，从相对容易改正的拼写错误到那些大错误，这使得几乎不可能识别用户的意图；（2）越是不严重的错误拼写越是经常发生；以及（3）正确拼写倾向于比错误拼写更经常发生。与 Albert Einstein 相关的搜索查询的统计量的一个例子如表 1 所示：

表 1：**Albert Einstein** 查询

搜索查询	频率
albert einstein	10135
albert einstien	1320
albert einstine	221
albert einsten	53
albert einstein's	52
albert einsteins	48
albert einstain	21
albert einstin	20
albert eintein	19
albeart einstein	17
aolbert einstein	11
alber einstein	8
albert einseint	7
albert einsteirn	7
albert einsterin	7
albert eintien	6
alberto einstein	6
albrecht einstein	5
alvert Einstein	5

在这个上下文中，可以针对拼写改正问题给出下面的迭代公式表示：

给定 $s_0 \in \Sigma^*$，找到一个序列 $s_1, s_2, \ldots s_n \in \Sigma^*$，使得 $dist(s_i, s_{i+1}) \leq \delta$ 且 $P(s_{i+1}) = \max_{t \in \Sigma^* : dist(s_i, t) \leq \delta} P(t)$，$\forall i \in 0..n-1$，以及 $P(s_n) = \max_{t \in \Sigma^* : dist(s_n, t) \leq \delta} P(t)$。

通过迭代地应用本发明的基本拼写检查程序进行改正的一个例子是：

anol scwartegger → arnold schwarzenegger

拼错的查询：	*anol scwartegger*
第一次迭代：	*arnold schwartnegger*
第二次迭代：	*arnold schwarznegger*
第三次迭代：	*arnold schwarzenegger*
第四次迭代：	没有进一步的改正；因此，输出第三次迭代的结果。

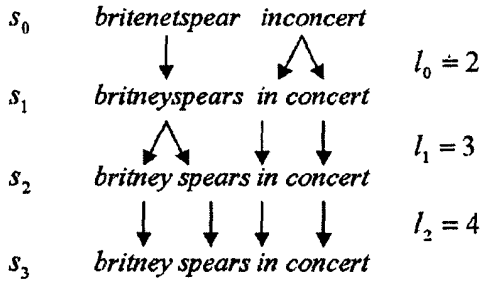
到这个时候，公式表示中提交给拼写检查问题的字符串的意思已经指定了。由 Brill 等人研究的一种可能（见，Brill, E.; Chandrasekar, R. 和 R. Rounthwaite, R 的美国专利申请公开说明书 US2003/0037077A1，申请号为 09/681,771；发明名称为“Spelling Correction System and Method for Phrasal Strings Using Dictionary Looping”）是把整个查询看成需要改正的字符串。他们的方法是建立一个统计字符错误模型，并以查询的相关频率与字符错误模型的一致性为基础，建立从已记入日志的查询到其他已记入日志的查询的映射，作为改正。他们迭代进行这一查询日志的改正过程，最后存储记录中被识别为记录中另一个查询的错误拼写的每个查询。在查询层次上进行工作有一些主要缺点。这种方法利用 web 查询日志中可以得到的大量信息，但是只覆盖了出现在这些记录中的正确短语匹配，具有相对较低的覆盖范围。像 *britnet spear inconcert* 那样的查询不能被改正，因为它的正确形式 *britney spears in concert* 并不出现在查询日志中，尽管它的子串可以被单独改正，例如：

britnet spear → britney spears

本发明克服上述方法的问题，创造出一种有效的查询拼写检查方法。本发明利用下面的公式表示：

给定 $s_0 \in \Sigma^*$，找到一个序列 $s_1, s_2, \ldots s_n \in \Sigma^*$，使得：对每个 $i \in 0..n-1$ 存在分解 $s_i = w_{i,0}^1 \ldots w_{i,0}^{l_i}$，$s_{i+1} = w_{i+1,1}^1 \ldots w_{i+1,1}^{l_i}$，此处 $w_{j,h}^k$ 为满足 $dist(w_{i,0}^k, w_{i+1,1}^k) \le \delta$，$i = 0..n-1, k = 1..l_i$ 的词或词组；对 $i = 0..n-1$，有 $P(s_{i+1}) = \max_{t \in \Sigma^*: dist(s_i, t) \le \delta} P(t)$；且有 $P(s_n) = \max_{t \in \Sigma^*: dist(s_n, t) \le \delta} P(t)$。

可以观察到，从一次迭代到下一次迭代，字符串分解的长度可以变化，例如：



迭代过程易于出现其他类型的问题。短查询能被迭代地转换成其他不相关的查询；因此，在本发明一个实例中，改变这样的查询还有另外的限制。为了进行训练，本发明除了使用可用的词典信息（例如，像 *multi-modal* 那样的辞典条目将不会被拆分为三个部分）之外，还利用非常基础的空格和词定界符信息来对 web 查询日志中使用的所有查询进行标记化，并收集一元模型和二元模型统计量。

利用与处理查询日志中所使用的相同空格和词定界符信息和可用词典信息，对输入查询进行标记化。对于每个标记（token），利用前面的字符串距离函数（在本发明一个实例中，该函数是所述的加权 Levenshtein 函数），对在辞典之内的词和在辞典之外的标记使用不同的阈值，计算出一组备选方案。在从词典和日志中提取出来的词/标记的一元模型和二元模型的空间中搜索匹配项。在本发明的一个实例中，一元模型和二元模型被存储在一个相同的数据结构中，以使得系统能以与处理单个词的未知形式完全相同的方式来处理词的串接和拆分。

一旦为查询中的每个词形计算出所有可能的备选方案组，就利用 Viterbi

搜索（其中通过利用二元模型和一元模型查询日志的统计量来计算转换概率，并用词之间的逆序距离来代替输出的概率），在下列约束下（例如）查找输入查询的最好的可能备选字符串：不允许同时改变两个毗邻的在词汇表之内的词。该约束避免了像 *log wood* → *dog food* 那样的改变。该约束的一个算法结果是不需要搜索网格图（trellis）中的所有可能路径（即是说，查询中的每个标记的候选改正的所有可能组合），这使得修正的搜索过程大大加快，如下面所进一步描述的那样。假设每个词的备选方案列表随机分类但有这样的特性：若输入的词形在可信辞典之内，则输入词就在该列表中的第一个位置上，那么被搜索过的路径就形成“边缘（fringe）”。在修正 Viterbi 搜索的示例 300 中，图 3 给出网格图的一个例子，该网格图中假设 w^1、w^2 和 w^3 为在辞典之内的词形。可以观察到，不是计算对应于备选方案 w^1 和 w^2 之间的 $k_1 \times k_2$ 个可能路径的成本，而是仅需计算 $k_1 + k_2$ 个路径的成本。

由于本发明使用词二元模型的统计量，停用词（如介词和连词）可能会多次消极地干扰最好的路径搜索。例如，在改正像 *platunum and rigs* 那样的查询时，基于词二元模型的语言模型将不使用与词形 *rigs* 有关的上下文。为了避免这个类型的问题，停用词及其最有可能的拼写错误被分别对待，首先忽略它们而进行搜索，如示例 300 中所示，此处 w^4 被假定为这种词。一旦在受限制的网格图中发现最好的路径，就由第二 Viterbi 搜索用极值固定的边缘计算停用词（或其错误拼写）的最好备选方案，如图 4 中停用词处理修正 Viterbi 搜索的示例 400 所描述。尽管由于太过于限制搜索空间，采用边缘的搜索方式似乎对准确度存在可能的负面影响，但是当与拼写改正的迭代过程相结合时，这种方式是非常强大的。

在 web 查询中估计 n 元模型时，本发明能使用查询日志和 web 索引，作为所提议的带有边缘方法的 Viterbi 搜索所需要的有价值资源。然而，如果独立使用，它们都不能被认为是比其他资源更好的资源。一方面，web 频率可能对迭代改正方式不甚有用，因为 web 文档错误没有 web 查询错误频繁（大约 10% 到 15%的 web 查询包含拼写错误），而且网页作者所犯的错误不能用来对查询 web 者所犯的错误进行很好的建模。同样，web 频率可能并非必然反映查询拼写改正中某些词的重要性。由于空间和速度的限制，出现在网络上的二元模型数目比拼写改正系统能利用的数目要大得多，因此只储存和利用比给定阈值值具有更高计数的二元模型可能不是最佳选择。相对于较低 web 计数但实际上却

出现在查询日志中的词二元模型如 *ox picture*，许许多多出现在网络上却不出现在查询日志中的二元模型如 *mail ox*，对查询改正来说可能比较不相关。

另一方面，对低计数的一元模型和二元模型来说，查询日志可能是相当不可靠的。例如，一个样本查询日志中有 20 个查询包含了二元模型 *catfish soap*，而只有 3 个查询包含了二元模型 *catfish soup*。基于这些统计量，像 *catfish sop* 那样的查询会与包含词 *soap* 的不正确备选方案联系起来（为简单起见，这个例子假设 $dist(sop,soap) = dist(sop,soup)$）。在本发明一个实例中，使用在查询日志中出现的词一元模型和二元模型，但同时根据它们的 web 频率调整它们的查询日志频率。这样，在获取更可靠的词 n 元模型统计量时，可以从查询日志中滤除频率非常低的不在 web 上发生的 n 元模型（这可以通过限制运行时中所用的数据大小来实现），而不丢失对迭代改正有用的频率较高查询的拼写错误。图 5 中，显示依照本发明一个方面的信息流结构 500 的示例。信息流结构 500 描述被用来重新估计查询日志特里结构（trie）504 的一元模型和二元模型统计量的 web 索引 502。

在本发明的其他实例中，可以通过充实可信辞典来增强辞典开发（尤其对非英语的其他语言来说），充实可信辞典的实现方法是在可信辞典之内增加经常通过拼写改正的词，如 *amd*：

amd processors → *amd processors*

amd warranty → *amd warranty*

amd overclocking → *amd overclocking*

在本发明又一个实例中，假定查询日志词直方图中有一个未知词，可以利用本发明来查找包含该词的所有查询，并计算其中词被改正的查询的数目和其中词没有被更改的查询的数目。在本发明的其他实例中，通过观察拼写错误在不同的上下文中如何改正，从查询日志提取出用户的知识。本发明的其他实例包括建造跨越不同语言的同源字典（在机器翻译中用作桥接元素）和/或使用迭代拼写改正程序促进语言翻译器。

考虑到上面所显示和描述的示例性系统和过程，参考图 6-8 的流程图将可以更好地理解依照本发明来实现的方法。

尽管出于简化解释的目的，所述方法被显示和描述为一系列的块，但应该

理解和明白，本发明不受块的顺序所限制，因为依照本发明的一些块可能以与在此所显示和描述所不同的顺序发生和/或与其他块同时发生。而且，实现依照本发明的方法并非需要全部例示的块。

本发明可以在像程序模块那样的计算机可执行指令的一般上下文中描述，由一个或多个组件执行。一般地，程序模块包括执行特定任务或实现特定的抽象数据类型的例程、程序、对象、数据结构等等。通常，程序模块的功能可以根据本发明各种实例的需要，进行组合或分配。

图 6 中，显示一个依照本发明一个方面的促进搜索查询的一种方法 600 的流程图。方法 600 开始（602），获得输入搜索查询（604）。查询通常由用户输入到 web 搜索应用程序内。于是本发明对搜索查询进行标记化，以便将其分解为处理所用的二元模型和一元模型（606）。然后，利用标记化的查询来计算备选查询建议，至少部分地使用至少一个查询日志（608）。查询日志至少部分地提供与用户所进行的先前搜索有关的统计信息。本发明可以使用这类诸如频率和/或改正建议等等的统计信息来提供备选查询建议。在本发明的其他实例中，可信辞典也可以和另外的 web 统计量一起，被用来改进低计数词 n 元模型。也可以只使用 web 统计量而不使用辞典信息。本发明也能利用当指点改变时允许字母的插入、删除、替换、毗邻调换和远距离移动等等的修正的上下文依赖的加权 Levenshtein 距离。在计算备选查询建议的过程中，本发明也可以使用不允许同时改变两个毗邻的在词汇表之内的词的约束。该约束的一个算法结果是不需要搜索网格图中所有可能的路径，这使得修正搜索过程大大加快。另外，它可以防止查询变成一个作为备选查询建议的完全错误的短语。一旦备选查询建议被计算出来，就输出给用户和/或系统（610），结束流程（612）。

参见图 7，描述另一个依照本发明一个方面的促进搜索查询的一种方法 700 的流程图。方法 700 开始（702），获得输入搜索查询（704）。于是，通过利用与处理查询日志中所使用的相同空格和词定界符信息和可用词典信息，对该输入查询进行标记化（706）。对于每个标记，通过利用加权 Levenshtein 距离函数，并允许对在辞典之内的词和在辞典之外的标记采用不同的阈值，计算一组备选方案（708）。本发明与传统的拼写改正形成鲜明对照，因其在从词典和记录中提取出来的词/标记的一元模型和二元模型的空间中搜索匹配项。在本发明一个实例中，一元模型和二元模型被存储在一个相同的数据结构中，以使得系统能以与处理单个词的未知形式完全相同的方式来处理词的合并和拆分。

一旦为查询中每个词形计算出所有可能的备选方案组，就用 Viterbi 搜索（其中通过利用二元模型和一元模型查询日志的统计量来计算转换概率，并用词之间的逆序距离来代替输出的概率）在下列约束下查找输入查询的最好的可能备选字符串：不允许同时改变两个毗邻的在词汇表之内的词（710）。该约束的一个算法结果是不需要搜索网格图中的所有可能的路径，这使得修正搜索过程大大加快，如下面所进一步描述的那样。于是确定是否已经找到最佳备选查询建议（712）。若是，就输出最佳备选查询建议（714），结束流程（716）。若否，最好的子串备选查询建议在迭代过程中再次被标记化（706），直到找到最佳的备选查询建议。

参见图 8，例示了又一个依照本发明一个方面的促进搜索查询的一种方法 800 的流程图。方法 800 开始（802），从逆序 web 索引获得低计数查询日志的 n 元模型的 web 统计量（804）。通过合并来自一个非常大的数据库的信息，增强 n 元模型的统计信息。然后至少部分地利用 web 统计量，作为用于迭代处理低计数查询日志 n 元模型的 n 元模型统计量（806），结束流程（808）。在本发明一个实例中，使用在查询日志中出现的词一元模型和二元模型，但根据它们的 web 频率调整它们的查询日志频率。这样，在获取更可靠的词 n 元模型统计量时，可以从查询日志中滤除频率非常低的不在网络上发生的 n 元模型（这可以通过限制运行时中所用的数据大小来实现），而不丢失对迭代改正有用的频率较高查询的拼写错误。

为了给实现本发明各个方面提供另外的上下文，图 9 和下列讨论提供对一个适当的计算环境 900 的简要而又一般化的描述，本发明各个方面可以在该计算环境中实现。尽管前面已经在运行在本地计算机和/或远程计算机上的计算机程序的计算机可执行指令的一般上下文中描述本发明，那些本领域内的技术人员将认识到，本发明也可以和与其他程序模块结合实现。一般地，程序模块包括执行特定的任务或实现特定的抽象数据类型的例程、程序、对象、数据结构等等。而且，本领域内的技术人员将会明白，本发明的方法可以与其他计算机系统配置一起实施，这些其他计算机系统配置包括单处理器或多处理器计算机系统、小型计算机、大型计算机以及个人计算机、手持式计算设备、基于微处理器的和/或可编程的消费性电子产品等等，其中的每个都可以与一个或多个关联的设备有效地进行通信。本发明的示例性方面也可以在分布式计算环境中实施，在分布式计算环境中，任务由通过通信网络连接的远程处理设备执行。然

而，本发明一些方面，如果不是全部方面，可以在独立计算机上实施。在分布式计算环境中，程序模块可以置于本地和/或远程的存储器设备。

本说明书中，术语“组件”意指与计算机有关的实体，可以是硬件、硬件和软件的组合、软件或执行中的软件。例如，一个组件可以是但不限于是，运行在处理器上的进程、处理器、对象、可执行程序、执行的线程、程序和计算机。作为示例，运行在服务器上的应用程序和/或服务器都可以是一个计算机组件。另外，一个组件可以包括一个或多个子组件。

参见图 9，实现本发明各个方面的一个示例性系统环境 900 包括一个常规计算机 902，常规计算机 902 包括一个处理单元 904、一个系统存储器 906 以及把包括系统存储器在内的各种系统组件耦合至处理单元 904 的系统总线 908。处理单元 904 可以是商业上可以买到的或专有的处理器。另外，处理单元可以实现为由多于一个的处理器构成的多处理器，例如可以是并行连接。

系统总线 908 可以是包括存储器总线或存储器控制器、外围总线以及局部总线在内的若干总线结构类型中的任一种，这些若干总线结构类型使用多种常规总线架构，诸如 PCI（外围组件互连）、VESA（视频电子标准协会）、微通道、ISA（工业标准结构）和 EISA（增强型 ISA 总线）等，仅举几个例子。系统存储器 906 包括只读存储器（ROM）910 和随机存取储存器（RAM）912。基本输入/输出系统（BIOS）914 存储在 ROM 910 中，它包含帮助在计算机 902 内的组件之间传输信息的基本例程，比如在启动过程中。

计算机 902 也可以包括，举例来说，一个硬盘驱动器 916、一个磁盘驱动器 918（例如，用于读取或写入一个可移动的磁盘 920）和一个光盘驱动器 922（例如，用于读取或写入一个 CD-ROM 盘片 924 或其他的光学介质）。硬盘驱动器 916、磁盘驱动器 918 和光盘驱动器 922 分别通过硬盘驱动器接口 926、磁盘驱动器接口 928 和光盘驱动器接口 930 连接到系统总线 908。驱动器 916—922 及其相关的计算机可读介质为计算机 902 提供数据、数据结构、计算机可执行指令等等的非易失性存储。虽然上面对计算机可读介质的描述中提到一个硬盘、一个可移动磁盘和一个 CD，但本领域内的技术人员应该意识到，可以被计算机读取的其他类型的介质，如磁带盒、闪存卡、数字视频盘、伯努利盒式磁带等等，也能在示例性的操作环境 900 中使用，更进一步，任何这些介质中都可以包含完成本发明的方法所用的计算机可执行指令。

若干程序模块可以存储在驱动器 916—922 和 RAM 912 中，包括一个操作

系统 932、一个或多个应用程序 934、其他程序模块 936 和程序数据 938。操作系统 932 可能是任何适当的操作系统或操作系统的组合。作为例子，应用程序 934 和程序模块 936 可以包括依照本发明一个方面的搜索查询拼写检查系统。

用户可以通过一个或多个用户输入设备往计算机系统 902 里输入命令和信息，如键盘 940 和定位设备 436（例如，鼠标 942）。其他输入设备（未示出）可以包括麦克风、操纵杆、游戏垫、卫星天线、无线电遥控、扫描仪等等。这些和其他的输入设备一般通过耦合至系统总线 908 的串行接口 944 连接至处理单元 904，但是也可以通过其他接口连接，如并行端口、游戏端口或通用串行总线（USB）。监视器 946 或其他类型的显示设备也通过一个接口如视频适配器 948 连接至系统总线 908。除了监视器 946 之外，计算机 902 可以包括其他外围输出设备（未示出），如扬声器、打印机等等。

应该明白，计算机 902 可以工作在使用到一个或多个远程计算机 960 的逻辑连接的网络化环境中。该远程计算机 960 可以是工作站、服务器计算机、路由器、对等设备或其他普通网络节点，而且通常包括与计算机 902 相关的上述元件中的许多或全部，尽管为简短起见，在图 9 中只例示一个存储器设备 962。图 9 中所描述的逻辑连接可以包括局域网（LAN）964 和广域网（WAN）966。这些网络环境常见于办公室、企业范围内的计算机网络、企业内部互联网和因特网。

当用于 LAN 网络环境时，例如，计算机 902 通过网络接口或适配器 968 连到局域网 964。当用于 WAN 网络环境时，计算机 902 通常包括调制解调器（例如，电话、数字用户线路 DSL、线缆 Cable 等等）970，或连接到局域网的通信服务器上，或有其他在 WAN 966 上建立通信的方法，如因特网。可以内置于或外置于计算机 902 的调制解调器 970 经过串行端口接口 944 连接到系统总线 908。在网络化的环境中，程序模块（包括应用程序 934）和/或程序数据 938 可以存储在远程存储器设备 962 中。应当明白，所显示的网络连接是示例性的，实现本发明的一个方面时，也可以使用在计算机 902 和 960 之间建立通信链路的其他手段（例如，有线的或无线的）。

依照计算机编程领域内技术人员的实践，已经参考由计算机（如计算机 902 或远程计算机 960）执行的动作和操作的符号表示法来描述本发明，除非另外说明。这些动作和操作有时被称为计算机执行。应该明白，动作以及用符号代表的操作包括处理单元 904 对代表数据位的电子信号的控制，其中作为结果，

数据位会引起电子信号表示的变换或转化，以及在存储系统（包括系统存储器906、硬盘驱动器916、软盘920、CD-ROM 924及远程存储器962）的存储位置中数据位的维护，由此配置或者改变计算机系统的操作和其他信号处理。维护这些数据位所在的存储位置是具有与数据位对应的特定的电、磁或光的特性的物理位置。

图10是另一个可以与本发明相互作用的一个样本计算环境1000的方框图。系统1000进一步例示了包括一个或多个客户端1002的系统。（诸）客户端1002可以是硬件和/或软件（例如，线程、进程、计算设备）。系统1000也包括一个或多个服务器1004。（诸）服务器1004可以是硬件和/或软件（例如，线程、进程、计算设备）。（诸）服务器1004可以主控线程执行转换，比如，通过使用本发明。在客户端1002和服务器1004之间的一种可能的通信可以以适合在两个或多个计算机进程之间传输的数据包形式进行。系统1000包括可以用来促进（诸）客户端1002和（诸）服务器1004之间通信的通信框架1008。（诸）客户端1002被连接到一个或多个可以用来存储（诸）客户端1002的本地信息的客户端存储器1010。同样地，（诸）服务器1004被连接到一个或多个可以用来存储（诸）服务器1004的本地信息的（诸）服务器存储器1006。

在本发明一个实例中，在两个或多个计算机组件之间传输的、促进搜索查询拼写检查的数据包至少部分地包含与搜索查询拼写检查系统有关的信息，该系统基于至少一个查询日志，给一组查询字符串至少部分地提供至少一个备选拼写。

应该明白，本发明的系统和/或方法可以用在促进搜索查询拼写检查的计算机组件和类似的非计算机相关组件中。此外，本领域的技术人员将认识到，本发明的系统和/或方法可以应用于一大批相关电子技术中，包括但不限于计算机、服务器和/或手持式电子设备等等。

上面所已经描述的内容包括本发明的例子。当然，出于描绘本发明的目的而描述每一个可以想到的组件或方法的组合是不可能的，但本领域内的普通技术人员应该认识到，本发明的许多进一步的组合和排列都是可能的。因此，本发明包括所有这些属于权利要求书的精神和范围内的改变、修改和变动。此外，在具体实施方式或权利要求书中用到术语“包含”的范围内，这样的术语是指以类似于术语“包括”的方式包括在内，此处的“包括”作为过渡词在权利要求书中使用时作出解释。

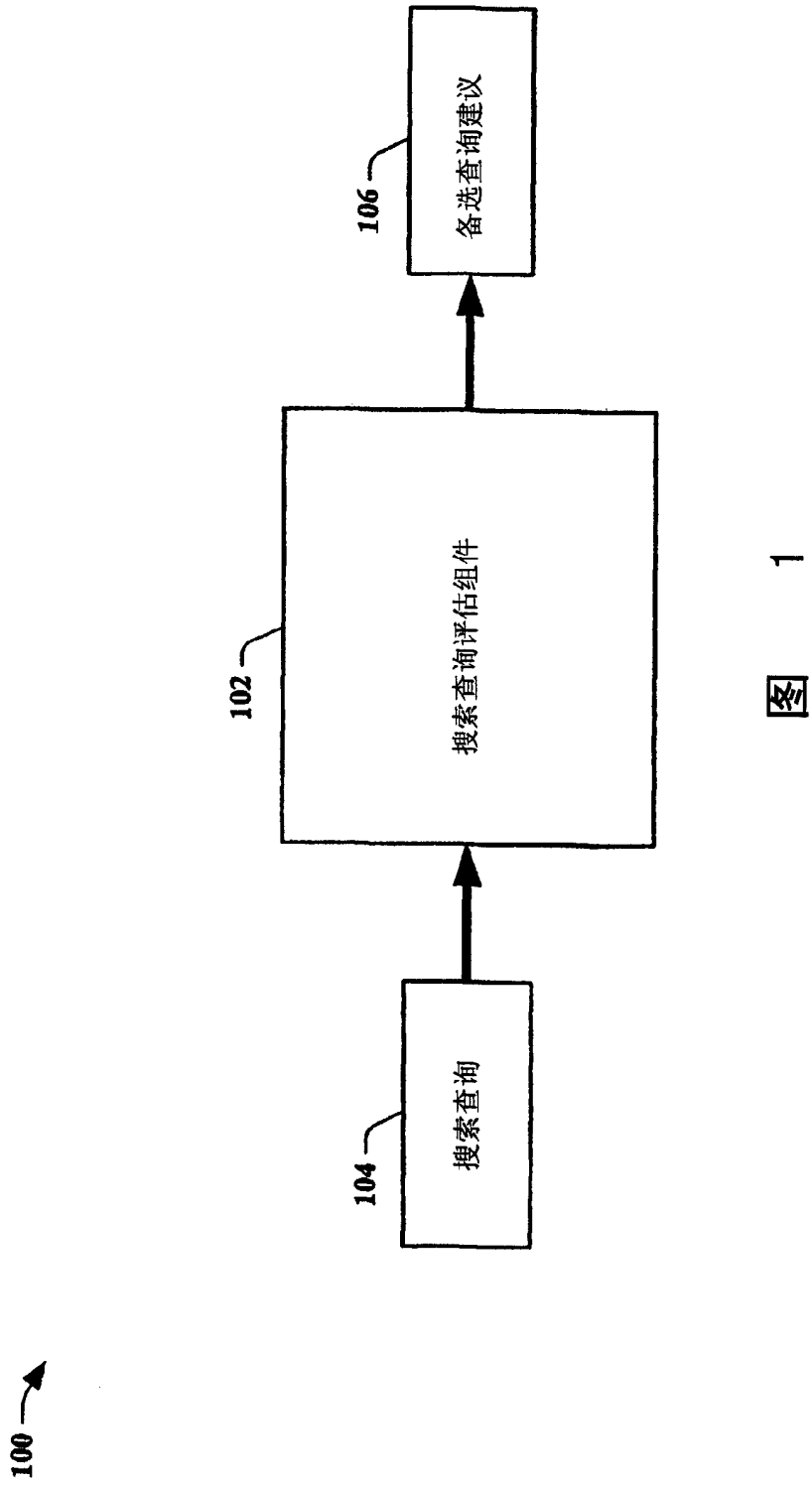


图 1

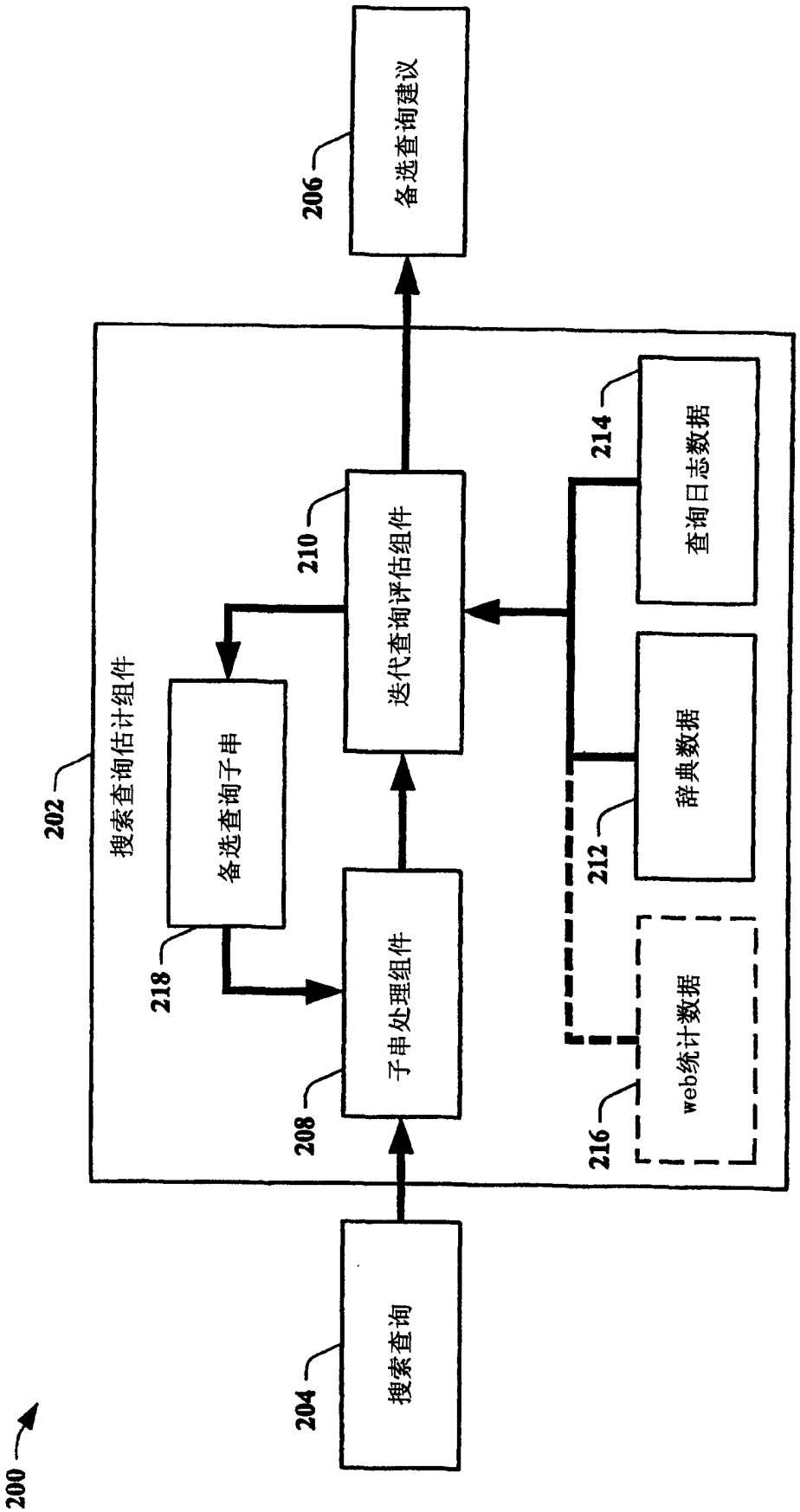

200
搜索查询
204
搜索查询估计组件
202
子串处理组件
208
备选查询子串
218
迭代查询评估组件
210
备选查询建议
206
web统计数据
216
辞典数据
212
查询日志数据
214

图 2

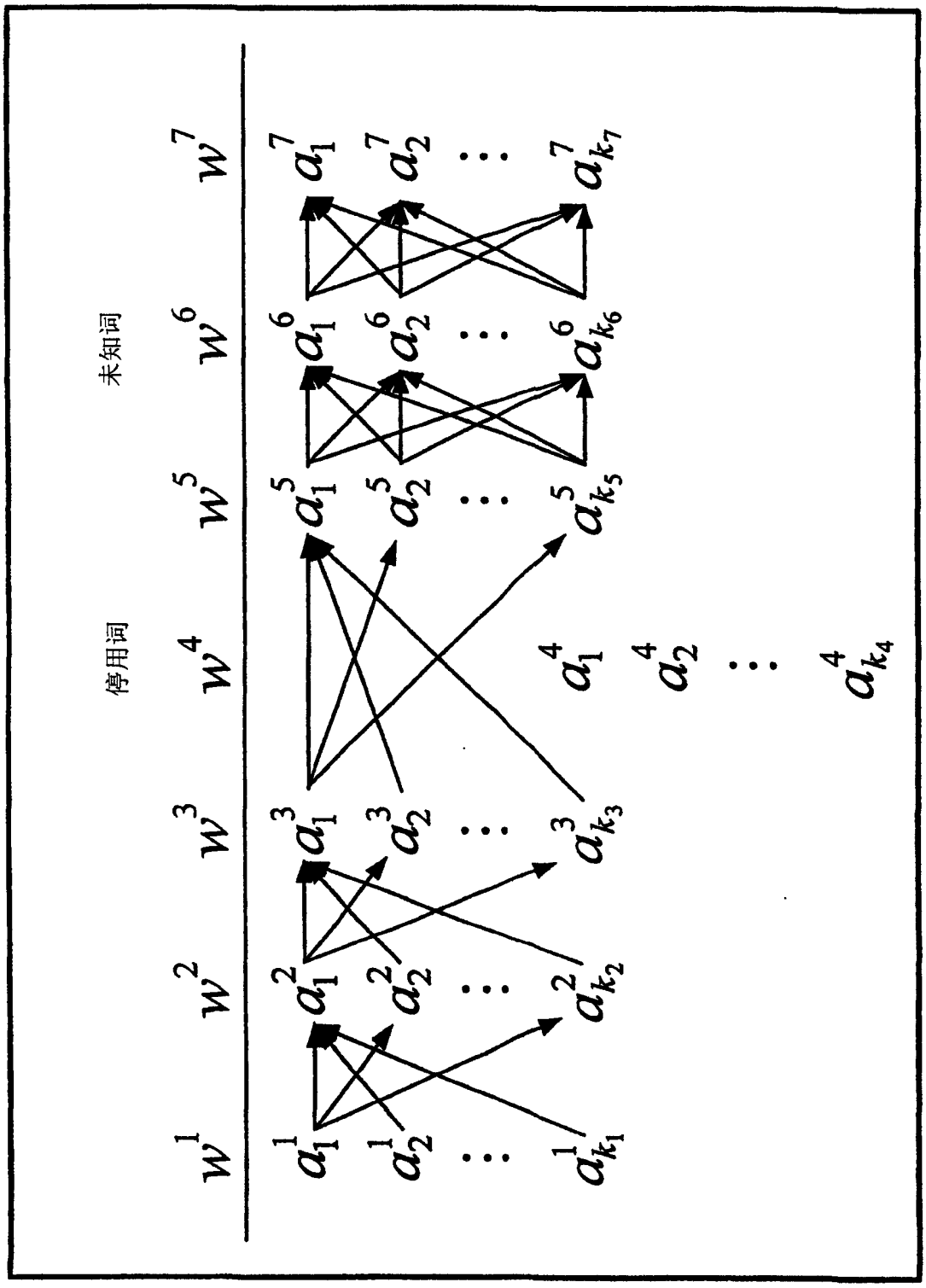

300
停用词
未知词
w^1 w^2 w^3 w^4 w^5 w^6 w^7
a_1^1 a_2^1 $a_{k_1}^1$
a_1^2 a_2^2 $a_{k_2}^2$
a_1^3 a_2^3 $a_{k_3}^3$
a_1^4 a_2^4 $a_{k_4}^4$
a_1^5 a_2^5 $a_{k_5}^5$
a_1^6 a_2^6 $a_{k_6}^6$
a_1^7 a_2^7 $a_{k_7}^7$

图 3

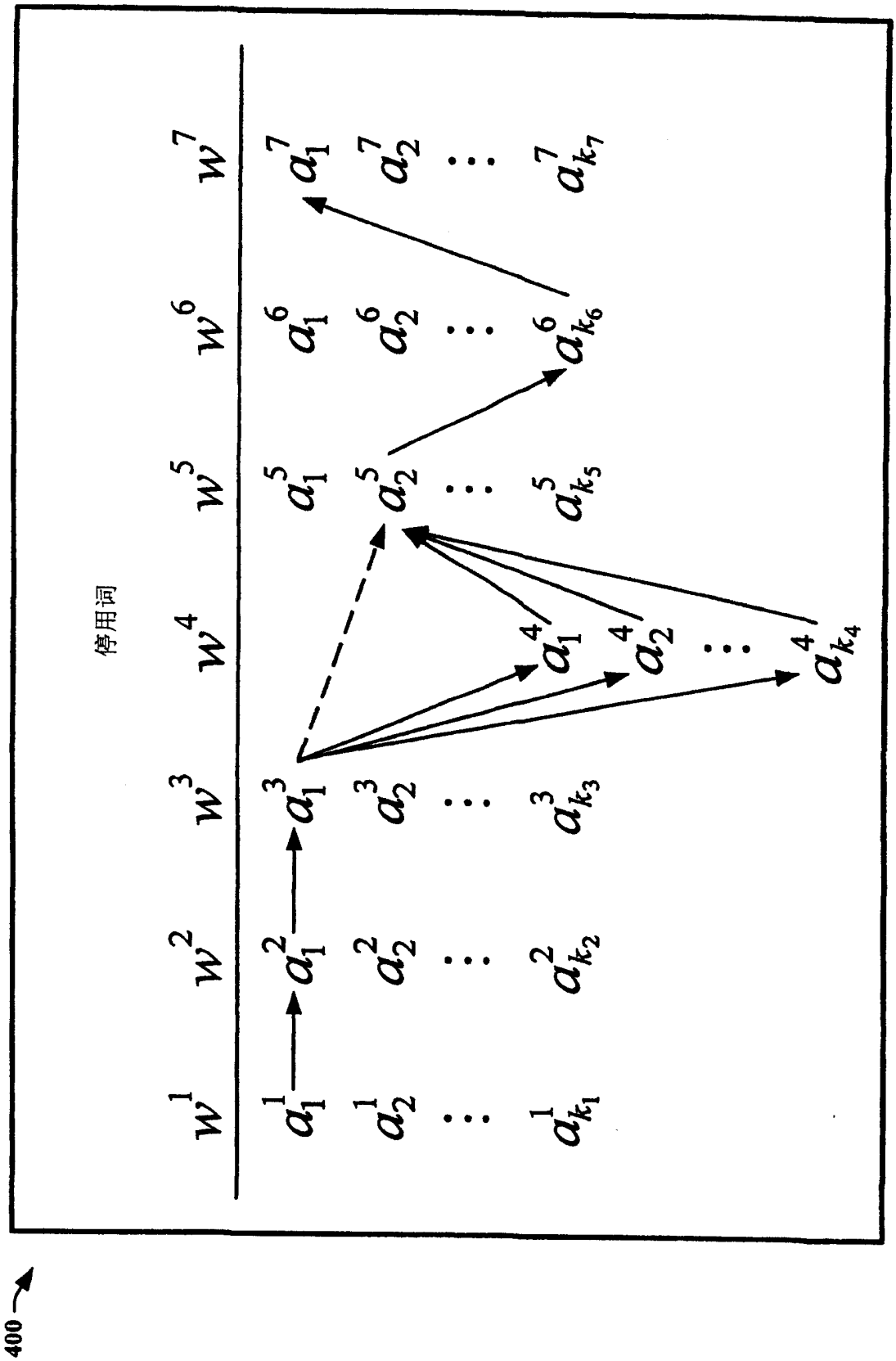

400
停用词
w^1 w^2 w^3 w^4 w^5 w^6 w^7
a_1^1 a_2^1 $a_{k_1}^1$
a_1^2 a_2^2 $a_{k_2}^2$
a_1^3 a_2^3 $a_{k_3}^3$
a_1^4 a_2^4 $a_{k_4}^4$
a_1^5 a_2^5 $a_{k_5}^5$
a_1^6 a_2^6 $a_{k_6}^6$
a_1^7 a_2^7 $a_{k_7}^7$

图 4

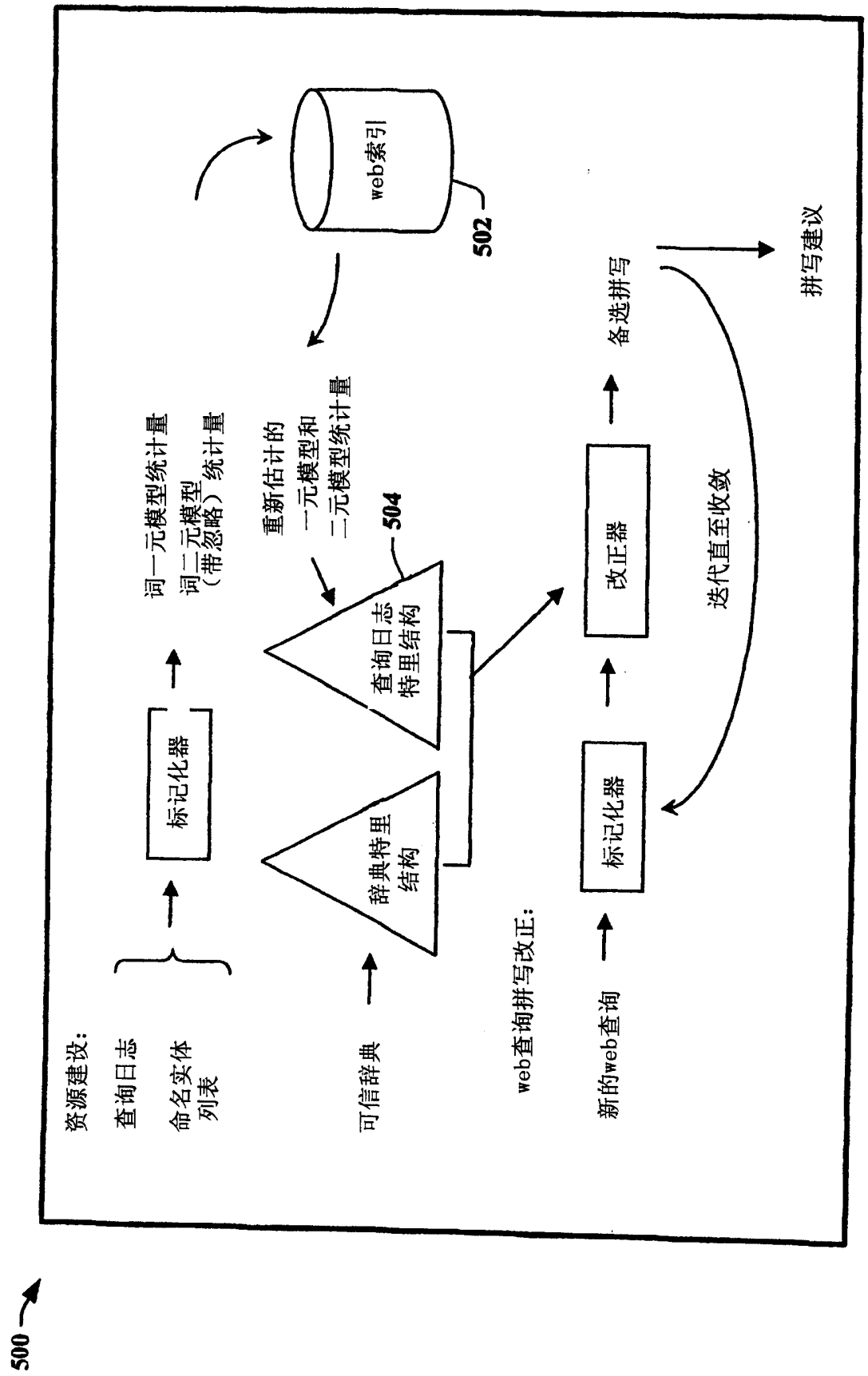

500
资源建设：
查询日志
命名实体列表
标记化器
词一元模型统计量
词二元模型（带忽略）统计量
web索引
502
重新估计的一元模型和二元模型统计量
504
查询日志特里结构
可信辞典
辞典特里结构
web查询拼写改正：
新的web查询
标记化器
改正器
备选拼写
迭代直至收敛
拼写建议

图 5

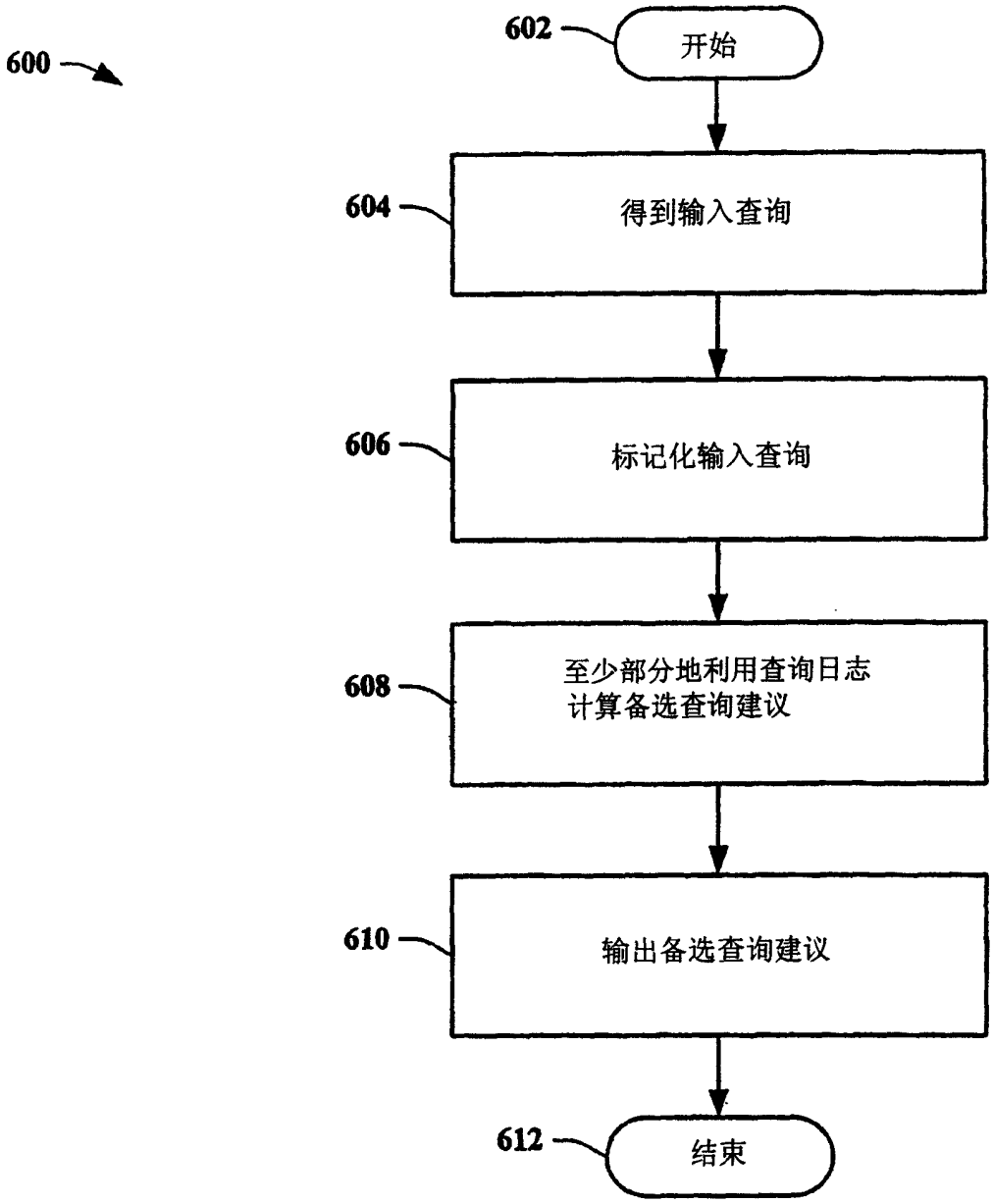

600
602
开始
604
得到输入查询
606
标记化输入查询
608
至少部分地利用查询日志
计算备选查询建议
610
输出备选查询建议
612
结束

图 6

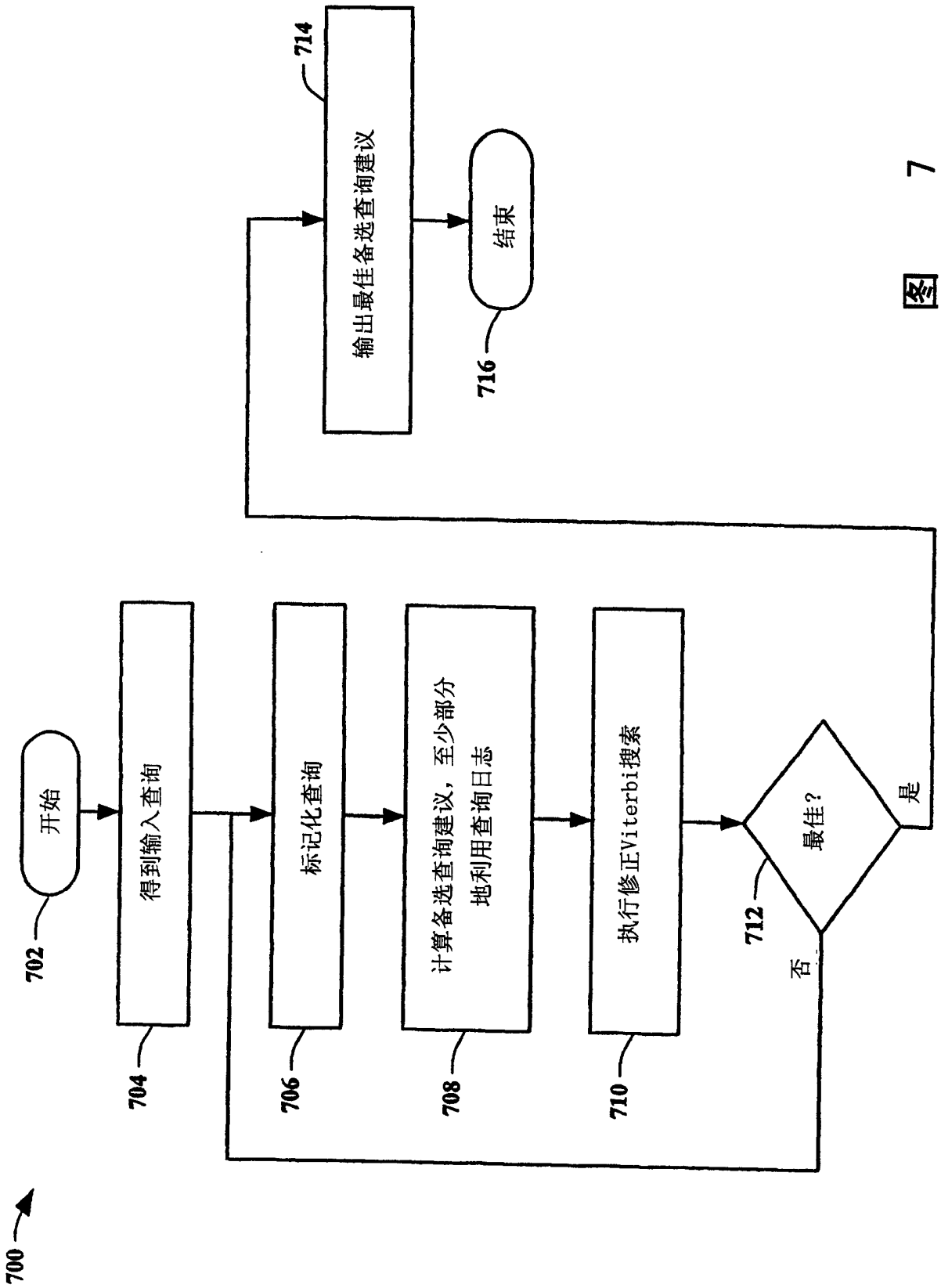

700
开始
702
得到输入查询
704
标记化查询
706
计算备选查询建议，至少部分地利用查询日志
708
执行修正Viterbi搜索
710
最佳？
712
否
是
输出最佳备选查询建议
714
结束
716

图 7

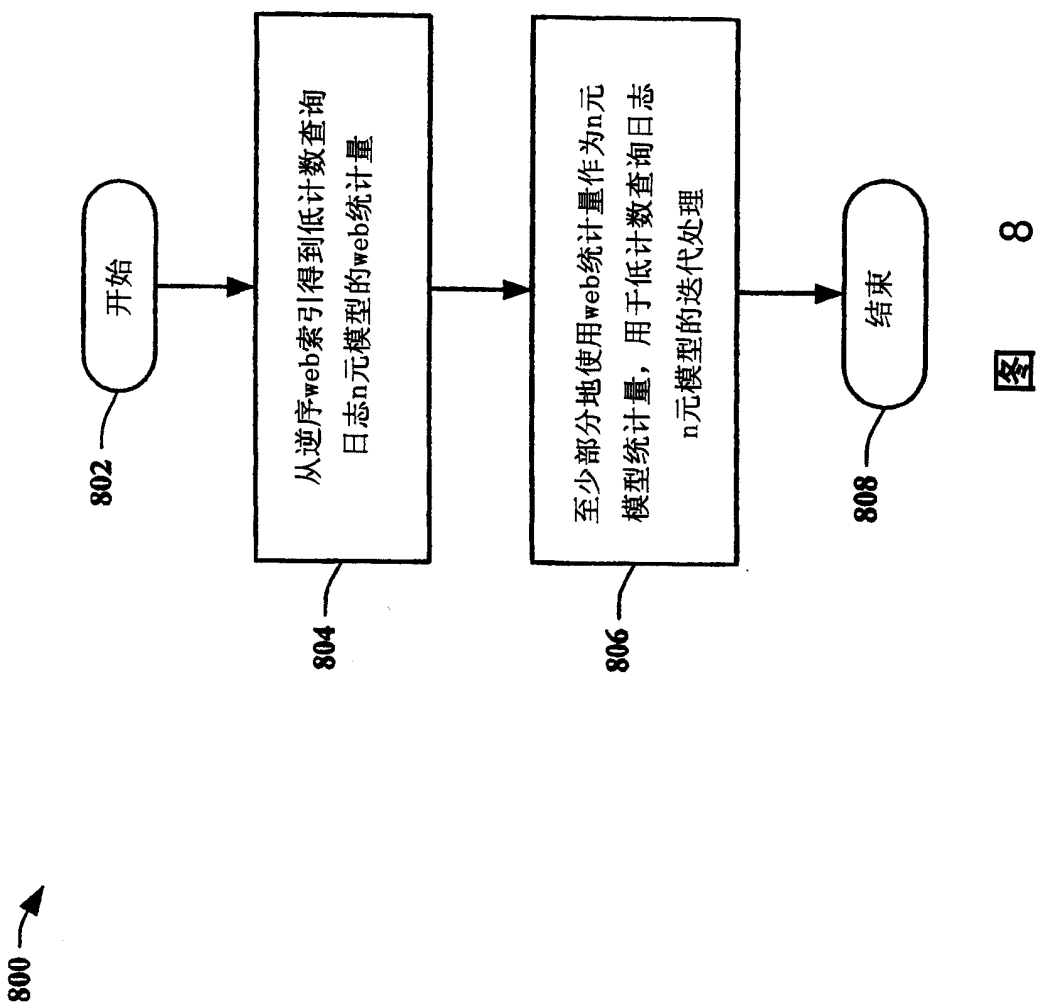

800
802
开始
804
从逆序web索引得到低计数查询日志n元模型的web统计量
806
至少部分地使用web统计量作为n元模型统计量，用于低计数查询日志n元模型的迭代处理
808
结束

图 8

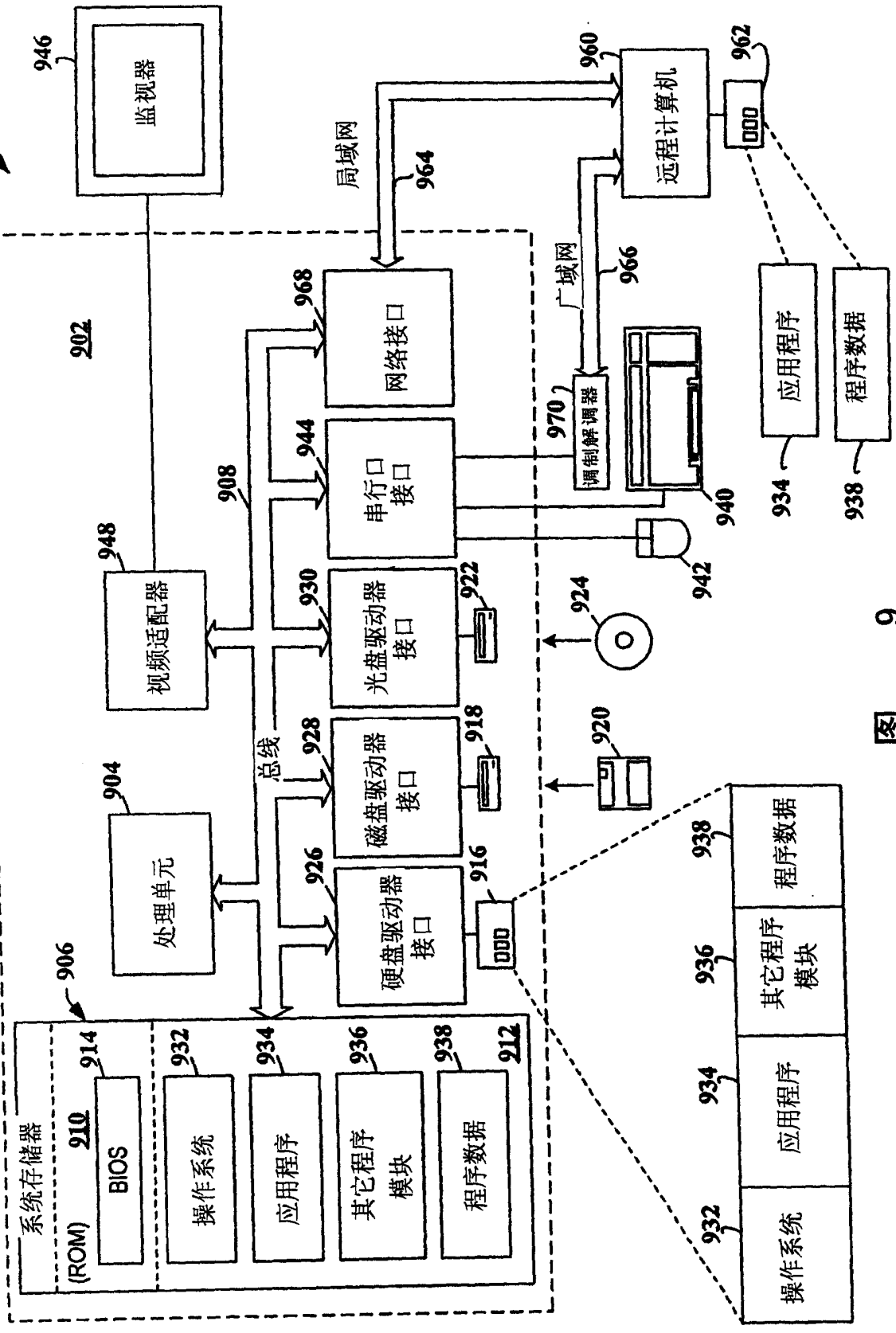

900
946
监视器
局域网
964
960
远程计算机
962
202
968
网络接口
广域网
966
970
调制解调器
944
串行口
接口
908
940
934
应用程序
938
程序数据
948
视频适配器
930
光盘驱动器
接口
922
924
942
总线
928
磁盘驱动器
接口
918
920
904
处理单元
926
硬盘驱动器
接口
916
906
系统存储器
(ROM)
910
914
BIOS
932
操作系统
934
应用程序
936
其它程序
模块
938
程序数据
212
936
其它程序
模块
938
程序数据
934
应用程序
932
操作系统

图 9

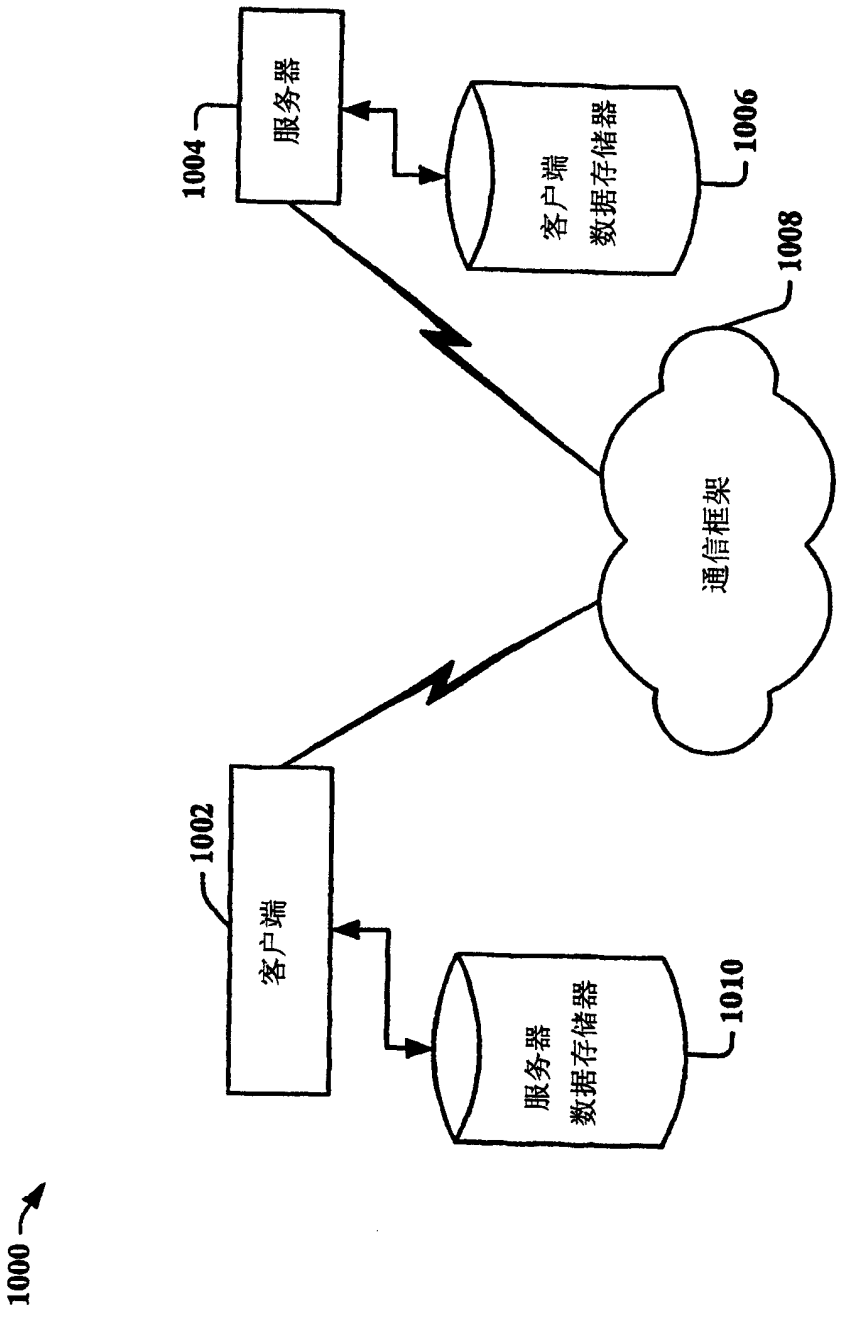

1000
1002
客户端
1004
服务器
1006
客户端
数据存储器
1008
通信框架
1010
服务器
数据存储器

图 10