第5章

用 户 交 互

本章学习目标

- 了解信息查询和查询提炼的知识。
- 掌握查询扩展和查询推荐的原理。
- 了解搜索结果页面展示相关知识。

从用户的角度来看，搜索引擎是用于提交查询和查看搜索结果的界面。用户与搜索引擎系统交互主要是通过构造查询、浏览检索结果以及重写查询的过程。同时这些交互过程也是信息检索过程中的一个关键部分，决定了搜索引擎是否提供了有效的服务。

5.1 信息需求与查询

搜索引擎能够帮助用户自动收集处理互联网上各种信息资源，并利用处理过的信息为用户提供服务。为了搜索引擎能提供高质量检索，需对用户查询意图进行准确识别。图 5-1 是搜索引擎系统结构示意图。

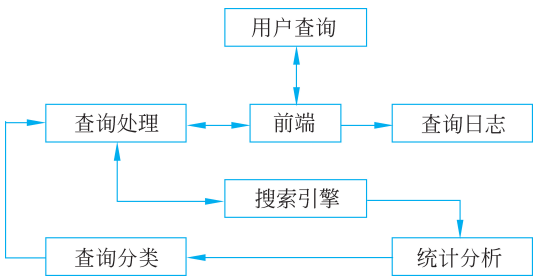

图 5-1 搜索引擎系统结构示意图

(1) 首先，搜索引擎系统通过前端与用户进行交互，当用户向搜索引擎提交查询后，若用户无附加保密要求则该前端会对查询词进行预处理，在搜索引擎查询日志中，对前端预处理后的数据提取分类特征。

(2) 然后，统计分析模块同时接收查询词和处理后获得的分类特征信息，并对这些信息进行统计分析，生成对应的特征向量。

(3) 接着,查询分类模块可用该特征向量判定用户的查询类别。

(4) 最后,搜索引擎接收处理后信息,并给出与之对应的搜索结果。若某用户对提交信息提出保密要求,则可能不会进行查询意图判断,此时前端处理后的查询信息将直接提交给查询处理模块进行相关处理。

信息需求是用户使用搜索引擎的动机。信息需求有不同的类型,研究人员根据不同的维度对这些信息需求进行归类,例如,需要查找的相关文档的数量、需求信息的类型、产生信息需求的任务等。对用户的信息需求进行分类,共分为以下 6 类。

(1) 咨询类:涉及用户想得到的建议、想法或解答等知识性查询,如用户通过搜索引擎输入关键词"什么是 PageRank?",希望获得 PageRank 的知识。

(2) 服务类:用户希望获得某个提供某种 Web 服务的网站。

(3) 导航类:用户不知道网站的 IP 地址或 URL,希望直接到达某个特定网站求助搜索引擎。

(4) 热点类:在查询日志中发现某些查询词具有明显的时间敏感性,在某特定时间出现频次非常高,属于热门搜索,特别是一些新闻性搜索。

(5) 学术类:用户在很多时候希望获得专业的文献、期刊、论文或书籍等信息,如用户希望检索 PageRank 的出处文献,输入查询信息"PageRank 的出处",如图 5-2 所示。这类查询本属于咨询类,但由于它们的信息非常专业和独立,而且相较于咨询类信息所占比例非常少,如不将这类信息独立分类,它们将淹没在其他咨询类信息的海洋里,对于很多需要获得相关学术性资源的用户将很难从搜索信息中获取其想要的资源。

图 5-2 学术类查询

(6) 资源类:用户希望在网络上获得软件、图片、音乐或视频等在线资源,或用户想将网络上某个资源下载到本地或某个设备上。

从搜索引擎设计者的角度观察这些信息需求,可以得出以下两方面的重要结论。

(1) 查询能够表达完全不同的信息需求,可能需要不同的搜索技术和排序算法来产生最好的排序结果。

(2) 查询仅仅是对信息需求的粗略表达。当用户发现难以表达出他的信息需求时,所提出的查询就会出现这种情况。然而,这种情况的产生通常是由于搜索引擎鼓励用户输入短查询导致的,这取决于搜索引擎的接口形式,而且过长的查询检索不到需要的结果。

目前在搜索引擎上最常用的查询形式,是采用由几个关键词组成的短查询,当前的搜索技术无法很好地处理长查询。例如,大部分互联网搜索引擎仅仅是将含有查询词的文档进行排序。如果一个人提交了一个包含 30 个词的查询,最可能的结果是什么都没找到。即使找到了包含所有词的文档,但在搜索结果中,往往也没有了那些符合语法结构的

长查询中表达出的隐含语言含义。搜索引擎上采用的排序算法主要是基于将文本看作词集合的统计学方法，而不是基于句法或语义特征。

当人们了解到采用长查询会得到什么样的搜索结果后，很快就认识到如何得到更可靠查询结果的方法，就是采用与他们要查找的信息相关的几个关键词去构成查询。但是这显然增加了用户的负担，本章即将描述的查询提炼技术，就是要减轻用户的负担，并且改善那些粗略表达的查询。

5.2　查询转换与提炼

5.2.1　停用词去除和词干提取

文本查询最初的处理过程应该对应于对文档的处理步骤，在查询文本中的词应转换成与文档文本处理时产生的词项相同的形式，否则在排序时会出现错误。在查询转换和文档转换中，仍存在许多的不同之处，特别是在停用词去除和词干提取处理上。对于其他一些处理步骤，如结构分析、词素切分，在查询中或者不需要这些处理，或者与对文档的处理方法基本上是相同的。

在索引文档时可以不去除停用词，而安排在处理查询的时候。在索引中保留停用词，可以增加系统处理含有停用词的查询的灵活性，对停用词的处理可以像普通词一样（留在查询中）。

在对英文分词时往往会采取词干提取，查询的词干提取是增加搜索引擎灵活性的另一项技术。如果在建立索引时对文档中的词进行了词干提取处理，那么查询中的词也必须进行词干提取。可是在有些情况下，对查询进行词干提取会降低搜索结果的准确性，例如，某些单词在不同的上下文中可能具有不同的含义，但在进行词干提取时被视为相同的词干。单词“meeting”可以指公司会议或两个人之间的会面，但在进行词干提取时会被处理为“meet”，导致搜索结果的混淆。如果索引文档的时候不进行词干提取，就能够在处理查询的时候决定是否对“meeting”进行提取词干。可以根据一些因素做这样的决定，例如，这个词是否是被引用的短语中的一部分。

为了使查询的词干提取能够获得更好的效果，一定要用恰当的词的变形来扩展查询，而不是将查询词减少到只剩下词干，因为并没有对文档进行词干提取。如果用词干“meet”代替了查询词“meeting”，查询就不再匹配包含“meeting”的文档。相反，查询应该扩展到包括词“meet”。查询扩展应该由系统（不是用户）根据某些形式的同义词操作去实现，或者也可以用词干和词来索引文档，这将使查询执行更有效，但是索引的规模也会增加。

每个词干提取算法都隐含着产生一些词干类别。词干类别是指能够通过词干提取算法转换成相同词干的一组词。这些词干类别的获取方法，是在一个大规模的文本集合上运行词干提取算法，并记录哪些词能够映射到给定的词干上。词干类别的数目可以十分庞大。例如，这里展示了三个词干类别（每个词干类别中的第一个词是词干）：

```
/bank banked banking bankings banks
```

```
/ocean oceaneering oceanic oceanics oceanization oceans
/polic polical polically police policeable policed
-policement policer policers polices policial
-policically policier policiers policies policing
-policization policize policly policy policying policys
```

在这些词干类别中,不仅包含较多的词("polic"类包含 22 个词),而且也包含一些错误。与"police"和"policy"相关的词,前者表示"警察",后者表示"政策",不应包含在相同的词干类别中,而且这也会影响排序准确率。尽管其他的词没有错,但可能会用在不同的上下文中。例如,"banked"更常用在讨论飞行或水池的时候,但是这个词干类别中的其他词,更常用在与金融相关的论述上。如果在扩展查询时直接使用词干类别,那么词干类别中词的数量也是一个问题。给一个简单的查询扩展了 22 个词,显然会影响系统的响应时间,并且如果没有正确地使用同义词操作符,可能会引起检索错误。

这两个问题能够通过文本集合中词的共现分析来处理,进行这样的分析是基于这个想法:能够互相替换的词的变形,应该经常在文本中共现。采用如下具体处理步骤。

(1) 对于词干类别中的每对词,计算它们在 W 个词的文本窗口中共现的次数,W 通常取 50～100 个词。

(2) 对于每对词计算共现或关联度指标,这用来衡量词之间的关联程度。

(3) 构造一个图,其中,顶点代表词,如果词共现的指标大于阈值 T,则用边连接它们。

(4) 找到这个图的连通分支,它们构成一个新的词干类别。

目前在一些应用(如网络检索)中,大量的查询日志也成为可利用的资源,利用这些资源的统计分析,可以验证提取的词干,甚至获取词干类别。对含有相同词的查询,从中分析趋向于共现的词语变形,这可以作为"fish/fishing"问题的一种解决方法,因为"fish"与"village"共现在一个查询中的可能性很小。

5.2.2 拼写检查

信息检索致力于为用户提供尽可能准确的返回信息,信息检索的准确性依赖于用户输入内容的准确性。当用户输入查询词串中包含错字、多字甚至少字时,信息检索系统的拼写检查能够尝试纠正用户输入的错误,向用户提供尽可能多的、正确的候选项,提高信息检索的准确率。因此,对用户输入内容进行拼写检查很有必要性。

信息检索在为用户提供搜索服务的同时,也依赖于用户带来的流量生存。但当用户输入内容包含错误时,信息检索系统若不具备拼写检查的功能,返回的信息可能会偏离用户的真实意图。这就降低了用户的体验度,造成用户流量的流失。长此以往,该系统就会被用户淘汰。因此,对查询内容进行拼写检查能够提高用户的满意度,给信息检索系统带来更多的用户流量,为信息检索更好的发展创造更有利的条件。

发展至今,英文文本拼写检查工作已取得了突破性进展。英文文本错误一般有两类:非词错误和真词错误。非词错误指的是一个词错用成非字典中的单词的现象,如 thsi cake is for you 中的"thsi"不在词典中,应改为"this"。另一类为真词错误,真词错误指一

个词错用成字典中另一个不符合当前上下文语义的词形成的错误，如“I come form Beijing”中的“from”被误拼成了“form”。图 5-3 是 Firefox 的拼字检查功能，这是输入“bluee”的结果。

图 5-3　Firefox 的拼字检查功能

在一些拼写检查工具中，采用的基本方法是对于在拼写字典中没有的词，就建议更正它们。将在字典中没有的词与字典中包含的那些词进行比较，并根据它们之间的一个相似度衡量标准，来提出更正建议。最常用的一个词之间比较的衡量标准是编辑距离，编辑距离是将一个词通过编辑转换成另一个词所需要的操作数。Damerau-Levenshtein 距离是指计算这个转换过程中单个字符插入、删除、替换、交换的最少次数。研究显示，80％或更多的拼写错误是由于这类单个字符错误引起的。

下面是 Damerau-Levenshtein 距离为 1 的一些词转换例子，它们只需要一个操作或编辑，就能够生成一个正确的词。

```
extenssions → extensions (插入型错误)
poiner → pointer (剔除型错误)
marshmellow → marshmallow (替换型错误)
brimingham → birmingham (交互型错误)
```

另外，在 doceration → decoration 的转换过程中，编辑距离为 2，因为有以下两步编辑操作。

```
doceration → deceration
deceration → decoration
```

下面来看如何计算编辑距离。假设两个字符串 string1 与 string2 的长度分别为 m 和 n，在计算编辑距离的时候需要构造一个二维矩阵 d，行数为 $m+1$，列数为 $n+1$，矩阵中的某个元素 $d[i][j]$表示 string1 的前 i 个字符与 string2 的前 j 个字符之间的距离，按照这种含义，那么 string1 与 string2 之间的距离就是 $d[m][n]$，这样计算编辑距离的问

题就转换为求解矩阵 d 的值。

对于矩阵 d 是知道第一行和第一列的值的,第一行的元素可以用 $d[0][j]$来表示,含义为 string1 的前 0 个字符与 string2 的前 j 个字符之间的距离,string1 的前 0 个字符其实就是空字符串,显然空字符串与任何其他字符串之间的距离都是该字符串的长度,因为只需要在空字符串中依次插入该字符串的每个字符就得到了该字符串,得到了 $d[0][j]=j$,同理可以得到 $d[i][0]=i$。以 string1="store",string2="shop"为例,矩阵 d 的初始状态如图 5-4 所示。

		s	h	o	p
	0	1	2	3	4
s	1				
t	2				
o	3				
r	4				
e	5				

图 5-4 矩阵 d 的初始状态

对于 d 中的任一元素 $d[i][j]$,表示 string1 的前 i 个字符与 string2 的前 j 个字符之间的编辑距离。考虑下面两个情况。

(1) 如果 string1 的第 i 个字符与 string2 的第 j 个字符相等,也就是 string1$[i-1]$==string2$[j-1]$,此时 string1 的前 i 个字符与 string2 的前 j 个字符之间的距离就等于 string1 的前 $i-1$ 个字符与 string2 的前 $j-1$ 个字符之间的距离,也就是 $d[i][j]=d[i-1][j-1]$,因为只需要把 string1 的前 $i-1$ 个字符变换成 string2 的前 $j-1$ 个字符,就得到了 string2 的前 j 个字符。

(2) 如果 string1 的第 i 个字符与 string2 的第 j 个字符不相等,string1 的前 i 个字符变换为 string2 的前 j 个字符的最后一步操作只可能是插入、删除和修改中的某一个,下面分别看这 3 种情况。

① 如果最后一步是插入,那么需要将 string1 的前 i 个字符变换为 string2 的前 $j-1$ 个字符,最后一步插入 string2$[j-1]$即可,这样至少需要 $d[i][j-1]+1$ 步,因为将 string1 的前 i 个字符变换为 string2 的前 $j-1$ 个字符至少需要 $d[i][j-1]$步,加上最后一步的插入操作,所以总共至少需要 $d[i][j-1]+1$ 步。

② 如果最后一步是删除,那么需要将 string1 的前 $i-1$ 个字符变换为 string2 的前 j 个字符,最后一步删除 string1$[i-1]$,至少需要 $d[i-1][j]+1$ 步。

③ 如果最后一步是修改,那么需要将 string1 的前 $i-1$ 个字符变换为 string2 的前 $j-1$ 个字符,最后一步将 string1$[i-1]$修改为 string2$[j-1]$,至少需要 $d[i-1][j-1]+1$ 步。

对于上面 3 种情况,只需要取步数最少的即可,因此得到递推式(5-1):

$$d[i][j]=\begin{cases} d[i-1][j-1], & \text{string1}[i-1]=\text{string2}[j-1] \\ \min\{d[i-1][j],d[i][j-1],d[i-1][j-1]\}+1, & \text{其他} \end{cases} \tag{5-1}$$

对于上面的例子,最终得到的矩阵 d 如图 5-5 所示。

图 5-5 矩阵 d 的计算过程大致如下,string1="shop",string2="store",当 $i=j=3$ 时,此时 string1$[2]$=string2$[2]$='o';因此 $d[3][3]=d[3-1][3-1]=d[2][2]$。当 $i=2,j=2$ 时,string1$[i-1]$=string1$[1]$='h',string2$[j-1]$=string2$[1]$='t',两者不相等,因此

$$d[2][2]=\min\{d[1][2],\ d[2][1],\ d[1][1]\}+1$$

一个拼写错误会有多个可能的更正形式,例如,拼写错误

		s	h	o	p
	0	1	2	3	4
s	1	0	1	2	3
t	2	1	1	2	3
o	3	2	2	1	2
r	4	3	3	2	2
e	5	4	4	3	3

图 5-5 矩阵 d 的最终状态

“lawers”可能会有下列编辑为 1 的更正形式：lawers→lowers、lawyers、layers、lasers、lagers。拼写校正程序要决定是否将所有这些词都展示给用户，并且以什么样的顺序展示。一个典型的策略是，按它们在语言中出现频率的递减顺序呈现。注意，这种处理方式没有使用拼写错误的上下文。例如，在查询“trial lawers”中出现的拼写错误，不会影响到建议更正词汇的展示顺序。在拼写校正过程中不考虑上下文，也使得在查询中的一些拼写错误被忽略了，因为这些拼写错误产生了另一个词。例如，在查询“miniature golf curses”时，显然是单个字符删除错误的例子，用户原本输入的查询应该是“golf courses”，但是其中的拼写错误产生了词“curses”，这个词本身是正确的，所以就不能检测到这个查询中的拼写错误。

采用“Did you mean…”方式的典型界面中，要求拼写检查器生成一个简单的并且最优的建议。这意味着查询的拼写检查与字处理的拼写检查相比，利用上下文和频率信息去排序更正建议是最重要的，在字处理拼写检查中的更正建议，可以采用下拉列表的形式。

5.3　查询扩展

目前，大部分搜索引擎都以关键词为基础，通过字符匹配来检索内容，但语言中普遍存在同义、歧义的词语，所以可能会出现检索不匹配的情况。为了进一步满足用户的信息需求，需要对查询扩展技术进行完善和更新。

查询扩展是解决用户查询和文档词语之间词不匹配的一个有效手段。它根据用户输入的初始查询语句，通过一些有效的扩展算法添加一些与用户初始查询相关的词语作为扩展词，使用扩展词集合去检索，从而使返回的结果中包含更多与初始查询相关的页面。

查询扩展技术通常是基于对指定的文档集中词或词项共现的分析，文档集可以采用全部的文档集合、大规模的查询集合，或者在排序结果中最高排序的文档集。

5.3.1　基于全局分析的查询扩展

1. 全局聚类

全局聚类，就是聚类文档集中的各个词语，根据聚类结果生成相应的簇，然后通过某种策略选取扩展词，达到查询扩展的目的。早在 1971 年，国外就有学者提出了聚类算法的概念，这种方式有一个假定基础，就是文档中有两个相关词语，因此有很大概率会在文档集中同时出现。所以，学者通过全局词语共现进行聚类，之后再进行查询扩展。然而，实验结果没有想象中成功。

随后对这种方法又进行了一段时间的研究，使该方法有了更强的适用性，促进了检索系统的改善，提升了检索性能。但该方法也有明显的缺点，即对歧义词的处理能力较差，如果查询内容的意义较多，则该方法难以进行聚类簇分配，进而导致检索结果比较含糊，从而会影响系统检索性能。所以，在技术研究的过程中，要合理应用该技术方法，做到扬

长避短。

2. 语义扩展

搜索引擎需要考虑不同知识域的信息差异,以及用户查询与扩展词之间的语义关系,克服训练数据集较小的困难,更加准确地表达和扩展查询以提高查全率和查准率。语义扩展查询方法是有效提升查询准确率与召回率的方法之一,其主要包括四个步骤:数据预处理,产生并排序候选特征,候选特征词选择,查询词重组,如图 5-6 所示。

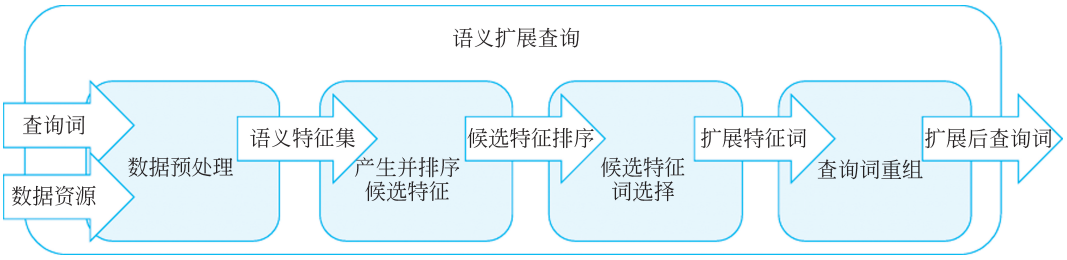

图 5-6 语义扩展查询步骤

语义概念查询扩展技术通过构建语义概念空间确定概念及概念之间的语义关系,实现同义词扩展、语义蕴含扩展、语义外延扩展和语义相关扩展。语义概念查询扩展技术能有效提高信息检索性能。


基于相关反馈与伪相关反馈的查询扩展

5.3.2 基于相关反馈和伪相关反馈的查询扩展

相关反馈是一种查询扩展和查询提炼技术,最初在 20 世纪 60 年代提出,依靠系统与用户的交互过程,识别出在用户初始查询的排序文档中的相关文档。在相关反馈中,让用户指出哪些文档是感兴趣的,以及哪些是完全不相关的。根据这些信息,系统增加词项或对原始词项重新分配权重,自动地改写查询,并用改写的查询生成新的文档排序。

通常的做法是,当一些词在相关文档中出现的频率比不相关文档或整个文档更高时,就将这些词增加到查询中,或者提高它们的权重。在伪相关反馈中,也是采用了相同的思想,但不是让用户自己去识别相关的文档,而是系统将排序靠前的文档假设是相关的。在这些文档中,频繁出现的词被用作扩展用户的初始查询,具体的做法也要取决于所采用的检索模型。由伪相关反馈产生的扩展词项是根据整个查询的,因为它们是在这个查询的排序靠前的文档中抽取出来的,但是扩展的质量是由排序靠前的文档有多少是实际相关而决定的。

下面通过一个例子来说明这个方法是如何实现的。图 5-7 是一个查询“New space satellite applications”的搜索排序结果,其中最左侧列是相关性评分。

用户使用“+”标记相关的文档,通过对这些文档进行全文分析,可以得到其中出现最频繁的词项以及相应的频率。其中,停用词不适合用作扩展的词项,它们不能表示出这些文档所包含的主题。改进这个过程的一个简单方法是,在文档的页面摘要中对词计数,并且去除停用词。通过这样的分析方法得到了下面这些高频词。

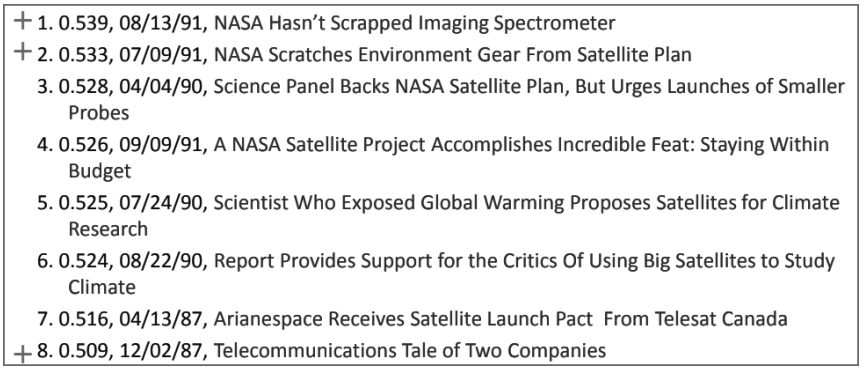

图 5-7　网页相关性排序结果

```
new, space, satellite, eos, launch, instrument, aster, application, nasa,
arianespace,bundespost,ss,rocket,scientist,broadcast,earch,oil,measure
```

这些词是用作查询扩展的更好的候选词项，使用这些词的主要作用是增加扩展词项的权重，图 5-8 是重新对这些词语赋的权重值。

2.074	new	15.106	space
30.816	satellite	5.660	application
5.991	nasa	5.196	eos
4.196	launch	3.972	aster
3.516	instrument	3.446	arianespace
3.004	bundespost	2.806	ss
2.790	rocket	2.053	scientist
2.003	broadcast	1.172	earth
0.836	oil	0.646	measure

图 5-8　扩展词项权重值

图 5-9 是经过相关反馈查询扩展，重新搜索“New space satellite applications”的搜索排序结果，与图 5-7 相比，原先排在第 2 名的文档，经过相关反馈查询扩展排在了第 1 位。

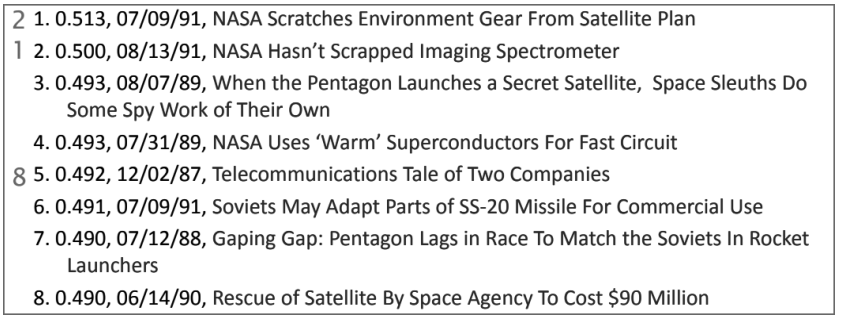

图 5-9　New space satellite applications 排序结果

相关反馈也可以用在其他应用中，如文档过滤。过滤涉及追溯一个人随时间而改变的兴趣，并且在有些应用中，允许人们使用相关反馈去调整他们的个人描述文件。另一个

相关反馈的简单应用是,在一些早期的搜索引擎中所采用的“more like this”功能,这个功能允许用户单击在搜索结果列表中与文档相关联的一个链接,这样可以产生与单击文档相似的另一个文档排序列表。这是一个相关反馈过程,但是对于训练数据仅限于一个相关文档。

伪相关反馈,也称为盲式相关反馈,提供的是一种自动局部分析方法,它是自动化相关反馈的手动操作部分,因此用户可不用参与额外的交互也可以获得更好的检索性能。这种方法首先通过普通检索从最相关的文档中寻找到一个初始结果,然后假定其中排名在前“k”的文档是相关的,最后在这个假设条件下像前面一样进行相关反馈。过程步骤如下。

(1) 把初始查询返回的结果当成相关结果(在大多数实验中仅前 k 个,k 为位于 10 和 50 之间的数)。

(2) 使用如 TF-IDF 权重的方法从这些文档中选择前 20～30(象征性的数字)个词语。

(3) 执行查询扩展,将这些词语加入查询中,然后再去匹配查询所返回的文档,最终返回最相关的文档。

一些实验,Cornell SMART 系统(Buckley et al.1995),在 TREC 4 实验环境中使用伪相关反馈提升了其检索系统的性能。这种自动化技术在大多数情况下都工作正常,有证据表明甚至好于全局分析。

通过查询扩展,一些在初始查询中错过的文档能被重新获得,从而提高了整体性能。很显然,这种方法的效果非常依赖于所选择的扩展词语的质量,目前已经发现它在 TREC 任务中提高了性能。但是,它并没有避免自动处理过程的危险。例如,如果需要查询的是铜矿,而且位于前面的一些文档都是关于智利的铜矿,那么在查询方向上会逐渐偏向于那些与智利有关的文档。此外,如果加入原始查询的词语与查询主题并不相关,检索质量有可能会下降,尤其是在 Web 搜索中,Web 文档经常会覆盖多个不同的主题。

5.3.3 基于查询日志的查询扩展

查询日志是用户在某一个搜索引擎上进行信息查询、信息浏览整个过程的记录。它包含一个用户从打开浏览器、进入某一个搜索引擎页面、输入查询词、返回结果、浏览页面,到关闭浏览器的整个用户的行为过程,所以,查询日志里面记录了很多用户查询行为过程的信息。通过分析这些信息,可以建立一个初始查询与用户文档之间的关联关系图,采用一些算法对用户文档打分、排序,从中选出相关性最高的前 n 篇用户文档作为相关用户文档集合。基于查询日志的查询扩展首先对查询日志和用户文件进行预处理,得到可以使用的查询日志结构和用户文档,并过滤相关用户文档集合,最后获取用户文档中词语和查询集合的相似度列表,选择排名前 m 位的词语作为查询扩展用词。图 5-10 是算法的总体流程图。