文章编号：1003-0077(2012)05-0033-07

基于逐点互信息的查询结构分析

朱亚东[1,2]，张 成[1]，俞晓明[1]，程学旗[1]

(1. 中国科学院 计算技术研究所，北京 100190；2. 中国科学院 研究生院，北京 100049)

摘 要：Web 搜索引擎中，对用户查询结构的有效分析，能更好地理解用户的查询意图，促进检索效果的提升。该文提出了一种简单高效的基于逐点互信息的查询结构分析方法，该方法包含了基于 MapReduce 的离线训练算法，以及一种自下向上的在线查询树构建算法。实验显示，该方法具有很高的切分速度，并能取得不错的可比较的切分效果。进一步的，该方法对检索性能的提升，也有明显的促进作用，在 MAP，p@5，p@10 评价指标上，都取得了不错的性能提升。

关键词：查询结构分析；MapReduce；在线查询树

中图分类号：TP391 **文献标识码**：A

Query Structure Analysis Based on PMI

ZHU Yadong[1,2], ZHANG Cheng[1], YU Xiaoming[1], CHENG Xueqi[1]

(1. Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100190, China;
2. Graduate University of Chinese Academy of Sciences, Beijing 100049, China)

Abstract: The effective analysis of user query structure is helpful for understanding the user's intent and promoting performance of the Web search engine. This paper proposes a straightforward and effective analysis method for user query structure based on PMI (pointwise mutual information). The method contains an off-line training algorithm based on MapReduce and a bottom-up online building method for query analysis. The experiment result shows that our approach possesses a high segmentation speed while maintain a comparable segmentation performance to other approaches. The experiment on TREC WT10g dataset further validates the effectiveness of our method and shows that it can prompt the search results in terms of MAP, p@5, p@10.

Key words: query structure analysis; MapReduce; online query analysis tree

1 引言

随着互联网的快速发展，SNS 社区、微博的快速兴起，互联网上的信息成爆炸式增长，用户对信息实时性、准确性的要求越来越高，相应的，对传统搜索引擎的考验也日益严峻。通常情况下，用户向搜索引擎提交的查询都是有一定含义的。所以搜索引擎必须对用户的查询进行有效的分析，这其中包括对用户查询结构的分析、用户查询意图的分析等，以期望能较好地满足用户的查询需求。同时，随着移动互联网的高速发展，移动终端在网络中的地位也日益凸显。在移动终端上，用户使用搜索引擎时，输入的查询词会更短，信息量会更少，这主要是由于移动终端输入不方便的限制。此时对用户查询的分析，就显得更加重要。本文主要针对用户查询结构的分析展开研究。

在对查询结构分析的过程中，为了能够更好的理解用户的查询意图，我们要尽量识别查询的语义信息，通常的做法是识别出查询中具有独立语义信

收稿日期：2011-09-21 定稿日期：2012-04-02

基金项目：国家自然科学基金资助项目(60903139，60873243，60933005)；国家 863 计划重点项目(2010AA012502，2010AA012503)

作者简介：朱亚东(1985—)，男，博士研究生，主要研究方向为互联网搜索与发掘；张成(1985—)，男，工程师，主要研究方向为信息检索；俞晓明(1977—)，男，高级工程师，主要研究方向为信息检索。

息、完整语法结构的短语子查询，进而指导后续的检索过程。查询结构分析主要关注查询中 term（分词之后的基本单元，即词项）之间的关联强度，主要体现在 term 之间共现作为词组短语的可能性。本文工作就是要找到一种合适的度量方法，度量 term 之间的关联强度。查询结构分析的结果是要产出一棵查询树，树的叶子节点是子查询，包括 term 查询，phrase 查询等。树中非叶子节点表示子查询之间的关系，主要有 AND，OR 关系，通常情况下，如果用户不做特别声明，默认的都是 AND 关系。本文讨论的方法，也是要构造一棵查询树，并且重点放在如何识别某一子查询为短语查询，指导后续的检索过程。本文提出了一套简单高效的基于逐点互信息的查询结构分析方法，其中包含了一种基于 MapReduce[①]的离线训练算法和一种自下向上的在线查询树构造算法。最后通过实验，验证了该方法的有效性，并取得了不错的检索效果提升。

本文的剩余部分组织如下：第二节介绍了一些主要的相关工作；第三节具体论述了基于逐点互信息的查询结构分析方法，包括离线训练算法和在线查询树的构造算法；第四节讲述了实验的设计与评估；第五节对文章进行了总结与展望。

2 相关工作

用户查询结构分析的主要目的是，识别查询中具有独立语义信息、完整语法结构的短语子查询，进而指导后续的检索过程。对外的表现就是如何对用户查询进行有效地切分，先后有学者从各个角度展开了研究。文献[1]是最早对 Web 查询的切分进行研究的文献之一。作者 Risvik 等人通过计算所谓的“连通性（connexity）”得分，对查询进行切分。这个“连通性”分数，是对一个切分片段（segment）内的互信息（MI：mutual information）和这个切分片段在查询日志中的频率信息的综合衡量。Jones 等人也使用了基于互信息的得分，去寻找那些内部相邻词项之间有较高互信息的切分[2]。此外，Kumaran 等人在关于长查询简化还原的工作中，也用到了点互信息对查询进行切分，进而寻找有意义的子查询串[3]。由此可见互信息是一种有效的查询切分方法，事实上，这也成为后续关于查询切分研究的重要基础。然而，不管是 Risvik 还是 Jones 都没有真正从实验的角度对他们方法的精确性进行有效地评估。

接着，文献[4]对标注数据中关于名词短语的句法和简单的语义特征进行学习，训练出一个线性组合的决策函数，进而对名词短语子结构进行了识别。该工作主要针对名词短语的特征展开了研究，对 Web 查询的切分并不具有合适的指导意义。文献[5]利用了一个公开语料库中的 n-gram 统计信息，引入了一种基于最大期望（EM：expectation maximization）的无监督切分模型。此外，作者还引入了 Wikipedia 的信息，进一步地提高模型切分的精度。文献[6]则在文献[5]工作的基础上，加入对用户点击的“查询文档对”信息的考虑，进一步地提高切分的精确性。文献[5]和文献[6]中的查询切分模型都具有较好的切分精度，但他们依赖的模型都很复杂，并且在应用过程中都需要利用最大期望对模型涉及的参数进行最优化估计，整个过程过于复杂繁琐，在实际的 Web 搜索系统中，面对用户的在线查询请求，显然不具有实用性。文献[7]试图提出一种简单直观的切分模型，与本文的思想较为类似，但其依赖于某个公开的 n-gram 语料库，并假设整个 Web 中的潜在短语结构都在这个语料库中出现，显然这是不合理的，并且作者并没有对这些数据进行平滑。同时文献[7]的方法虽然相对文献[5]较为简单，但其仍然包含了大量的指数运算和累加和运算，在计算复杂性上仍偏高。

本文在 MI 方法的基础上，提出了一套简单实用的查询切分方法。一方面提出了一种高效的基于 MapReduce 的离线训练算法，能快速有效地对大规模的语料数据进行训练，对 MI 方法中用到的 n-gram 信息进行有效提取，保证了 MI 方法在实际应用中的有效性；另一方面，本文基于逐点互信息（PMI：pointwise mutual information）的切分方法，通过计算两两词项之间的互信息，进而对多个词项间的互信息进行简单有效的估计，从而对查询中潜在的短语子结构进行有效的识别，指导后续的检索过程。

在文献[3]的工作中，对 n-gram 的语言模型进行了较为详细的研究，并得出如下结论：锚文本的语言模型和查询的语言模型更接近，其次是标题的语言模型；语义信息最完整的是正文语言模型，其次是标题语言模型。这对我们离线训练数据的选取有一定的指导意义。例如，在本文实验中，从语义完整性的角度出发，选取了语料数据的正文进行训练。

① http://labs.google.com/papers/mapreduce.html

另外，文中的统计结果认为，trigram 语言模型，在 unigram、bigram、trigram、four-gram 中，是效果最好的语言模型。然而本文在 MI 的基础上，主要利用 unigram 和 bigram 语言模型进行查询结构的分析，一方面是出于简单高效的考虑，利用两两词项间的互信息对多个词项间的互信息进行快速的估计；另一方面，文献[8]中的研究表明，Web 检索中，平均每个查询包含 1.5～2.6 个词项，显然利用 unigram 和 bigram 语言模型作为查询分析的基础，也有其合理性。此外，如果采用更长 n-gram 模型，也会面临很多由于数据稀疏(data sparseness)带来的问题。

另外，在针对汉语分词的研究中，一部分的学者也提出了双字耦合度的概念，本质上与 MI 的思想类似[9]。本文主要针对英语的 Web 查询结构分析进行研究，故在此不做详述。

3　基于逐点互信息的查询结构分析

本节首先对逐点互信息的方法进行了介绍与分析。然后在其基础上，提出了一种简单有效的查询结构分析算法，主要包含两部分内容：离线训练的算法和在线查询树的构造算法。对于离线训练部分，我们采用了目前比较流行的 MapReduce 算法对 unigram、bigram 语言模型进行训练，大大加快了训练的过程，缩短了训练周期，同时也能满足大数据量的训练场景。在线查询树构造部分主要是对用户查询进行拆分，逐对计算 PMI，根据 PMI 值大于阈值 θ 与否，判断是否拆分，进而构造查树。

3.1　逐点互信息的查询结构分析

PMI 是指已知两个离散变量的分布，求这两个变量联合分布的方法。在本文的应用场景中，t_1 的出现作为第一个变量，而 t_2 的出现作为第二个变量。二者共同出现的概率越大，成为词组短语的可能性就越大。PMI 的计算公式如下：

$$PMI(t_1,t_2)=\log\frac{p(t_1,t_2)}{p(t_1)p(t_2)}=\log\frac{p(t_2|t_1)}{p(t_2)} \quad (1)$$

PMI 具有以下的性质：

1) 对称性

$$PMI(t_1,t_2)=PMI(t_2,t_1) \quad (2)$$

2) PMI 的值有正有负，0 表示 t_1 和 t_2 是完全独立的，在本文的应用场景中，可理解为 t_1 和 t_2 组成一个词组的可能性比较低。正值表示组成词组的可能性较大，负值表示拆开的可能性非常大。

3) 当 t_1 和 t_2 总是作为词组出现的时候，$P(t_2|t_1)=1$，那么 PMI 的值为 $-\log(p(t_2))$，当 PMI 的值越接近于 $-\log(p(t_2))$，则表示 t_1 和 t_2 组成词组的可能性越大。

本文采用 unigram 和 bigram 的语言模型信息，较为精确地计算二者之间的关联强度，然后进一步地去判断他们作为词组短语内容的可能性。在实验的过程中，本文采用 θ 作为分界点，即相邻两个词作为词组的临界阈值。PMI 大于 θ 则认为是可以作为一个词组的查询，PMI 小于 θ 则认为应该拆分为两个子查询。一个最直观的选择是 $\theta=0$。但是从实际语料情况来讲，以 0 作为分界点，可能并不能满足实际训练语料的语言特征。这个临界值，需要我们在训练的过程中不断的调查，以找到最优的取值。

3.2　基于 MapReduce 的离线训练算法

这部分的算法主要是针对训练语料，训练语言模型。训练的方式采用基于 MapReduce 的框架，利用并行的方式加速训练。可以满足大规模数据的训练和高性能的需求，具体的算法流程描述如下。

1) 训练 unigram 语言模型，图 1 是对应的算法描述过程。

```
1:  ReduceNum=C * cluster_tasknode_number;
2:  Conf.setNumReduceTasks(ReduceNum);
3:  FileInputFormat.addInputPath();
4:  Map(){
5:        whitespaceAnalyzer;
6:        ToLowerCase;
7:        Output.collect(word,1);
8:  }
9:  Reduce(){
10:     While(values.hasNext)
11:         Sum+=values.next().get();
12:     InsertToVoldemort(word, sum);
13: }
```

图 1　一元语言模型的训练算法

其中，1 到 2 步是设置 reduce 的数目，这里 C 是一个常数，且 $0.95<=C<=1.75$。第 3 步是设置文件输入路径，即待训练数据的路径。4 到 8 步是 map 的过程，分别包含：按空格分词(5)，将单词转化为小写形式(6)，进行 map 阶段的输出(7)。9 到 13 是 reduce 的过程，主要是将每个 word 出现的次数累加起来(11)，然后将 word 和 sum 以对应的

key-value 形式插入到 voldemort[①] 中。其中，voldemort 是一种高速的 key-value 非关系型数据库，具有很高的随机读取性能，可以有效地保证在线算法的执行。

2）训练 bigram 语言模型，方式与 unigram 语言模型基本类似。不同之处在于 map 过程中的分词方式。Unigram 是按空格进行单字分词，作为 Key，而 bigram 训练则是将切分后的连续两个字然后作为 Key。例如，I love China，在 bigram 训练的 Map 阶段，切分的结果为：I，I+love，love+China。

3）对训练的语言模型进行平滑。在语言模型的理论中，平滑是要“劫富济贫”，在训练语料中出现频率为 0 的 unigram 或者 bigram，在更广泛的语料中，不一定是 0。对语言模型进行平滑的方法有很多，本文在实验过程中，使用了一种简单的平滑方法：加 1 平滑。

最后，总的训练框架可如图 2 所示，较为清晰的展示了这部分算法的主体框架。

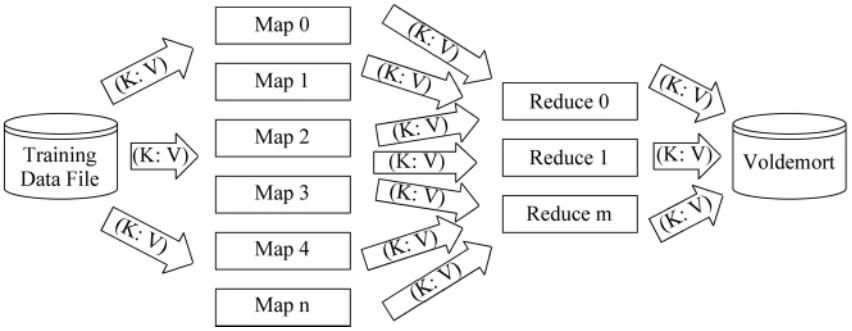

图 2 基于 MapReduce 的 Unigram 训练流程

3.3 在线查询树构造算法

这部分算法的主要功能是接受用户提交的原始查询，计算逐点互信息，对查询进行拆分，然后构造查询树。在线查询树构造的算法过程描述如下。

1）对用户提交的查询进行分词，英文按空格分词，并去掉停用词；

2）将分词的结果两两组合为一个 $Pair$，计算每一个 $Pair$ 的 PMI；

3）对于 PMI 小于 θ 的 $Pair$，以此为界，将查询拆分为 AND 关系的两个子查询；

4）如果 PMI 大于 θ，则将 $Pair$ 作为一个词组。如果 AB、BC 计算得到的 PMI 都可以作为词组，那么我们认为 ABC 就是一个词组；

5）根据 3）、4）步骤得到最终的查询树；

总体上，我们认为多个词项短语的 MI 是建立在 PMI 基础上的。例如，一个查询 ABC，如果 ABC 是作为一个短语存在，并在语料库中经常出现，则显然它的子集 AB、BC 也会经常共现，并且会有 ABC 共现的概率 $P(ABC)<=P(AB)$ 以及 $P(ABC)<=P(BC)$。所以对于 ABC 的 MI 计算，我们采取的方式为：$PMI(ABC)=Min(PMI(AB), PMI(BC))$。这也是我们算法中第四步的基础。

这是一种自底向上构建查询树的过程，通常情况下，查询树会有两层，叶子节点为 Term 查询和 Phrase 查询。对于某些长查询，以及查询扩展的情况来说，查询树会有多层。这些情况，我们会在下一步工作中进行研究。

下面本文使用 TREC 的查询测试集中的一个查询为例：newport beach California。这是一个地名的查询，在这个例子中，假定 $\theta=0$。按照算法的步骤来构造查询树：

1）分词结果为 $newport$，$beach$，$California$。

2）两两组合的 $Pair$ 为：$newport\ beach$，$beach\ California$。

3）计算这两组 $Pair$ 的 PMI：

a）$PMI(newport, beach)=2.54$，

b）$PMI(beach, California)=-3.39$。

4）根据算法步骤 4）的描述，应该将查询从 $beach\ California$ 中间拆分，默认作为 AND 关系子查询。$newport$ 和 $beach$ 的 PMI 大于 θ，故应该作为一个词组出现。

5）根据以上的计算，构造查询树如图 3 所示。

在实际应用中，这部分的算法必须具有很高的性能，并且能作为独立的服务考虑，以满足在线查询服务的需求。以往关于查询切分的研究通常都很复杂繁琐，在实际的应用中，显然是不可接受的。而本文的方法简单高效，具有很好的实用性。

① http://www.project-voldemort.com/

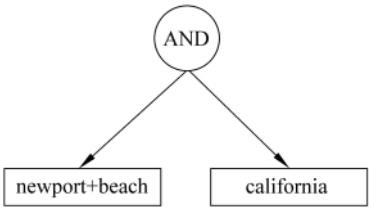

图 3　查询结构分析树

4　实验评估

4.1　数据集与实验设置

实验采用 TREC WT10g 的测试数据集作为训练语料，训练语言模型。使用基于 MapReduce 的离线训练算法对 unigram 和 bigram 进行训练，并将结果存入到 Voldemort 当中，以备实时使用。训练语料的大小是 10G，共 169w 左右的英文网页源码。本文仅仅对网页的正文(content)部分进行训练，没有对网页的标题(title)进行训练。这是因为，我们的任务是识别出具有独立语义的词组单元。而从语义完整程度的角度来看，content 的语言模型更具有代表性，更加能够体现语义上的特征[3]。

对于查询测试集，我们采用 Bergsma 等人建立的一个标准的查询切分数据集合[4]，作为我们的标准测试集合。文献[5]和[7]的实验评估中也都采用了这个查询测试集合。

4.2　查询切分实验

该章节实验主要用来验证本文提出的基于逐点互信息的查询结构分析方法的有效性，并与文献[5]和[7]的切分方法进行对比评估。

表 1 包含了在查询测试集上的测试结果。[5]和[7]分别对应文献[5]、[7]中的查询切分方法，最后一栏是本文的切分方法。对于这部分实验，我们只采用了标注者 A 标注的查询集合，它包含 500 个标注的训练集(Train set)，500 个标注的开发集(Development set)，500 个标注的测试集(Test set)。这些测试集中的查询都来自 AOL 2006 的查询日志。我们利用训练集和开发集调整选取本文方法的临界阈值 θ，以获得最优解。最后，整个查询切分实验都是在测试集上进行。对于实验的评估，我们采用了三类评估标准：(1)查询的精确性(query accuracy)：实验预测的查询切分与标准的测试集合完全匹配的比例；(2)分类的精确性(classify accuracy)：每两个连续的词项之间，正确预测边界的比例；(3)切分的精确性(segment accuracy)：预测的切分分段与标注测试集的匹配程度，采用了信息检索中的三个评价标准：分段的准确率(segment precision)，召回率(segment recall)，和 F 值(F-measure)。

表 1 中的结果显示，我们的方法在查询的精确性上与文献[5]和文献[7]中的方法分别差大约 0.02 和 0.03。在分类的精确性上也分别差 0.02 和 0.01，在切分精确性的三个指标上，也都有类似的差距。显然，这个差距并不是很显著(控制在 5% 以内)。事实上，如同我们在相关工作中提到的那样，用户查询结构的分析只是整个在线检索过程中的一部分，用户查询的在线响应速度是必须考虑的一个重要指标，这也是本文工作的一个重要出发点。文献[5]的方法是建立在一个较为复杂的生成语言模型之上，并且需要通过利用最大期望对模型涉及的参数进行最优化的估计，整个过程过于复杂烦琐，这在实际的 Web 搜索系统中，显然是不可接受的。文献[7]方法的出发点与本文类似，但其完全依赖某个公开的 n-gram 语料集，并假设整个 Web 中的潜在查询短语结构都在这个语料库中出现，显然是不可取的。同时文献[7]的方法虽然相对文献[5]较为简单，但相对于本文的方法，其仍然包含了大量的指数运算和累加和运算，在计算复杂性上还是高于本文的方法。

表 1　标准测试集上的切分性能

Annotator	Measure	文献[5]	文献[7]	Our Model
A	Query accuracy	0.526	0.536	0.503
	Classify accuracy	0.810	0.807	0.798
	Segment precision	0.657	0.665	0.602
	Segment recall	0.657	0.708	0.674
	Segment F	0.657	0.686	0.636

事实上，我们接着对本文方法的切分速度进行了简单测试。将离线训练出的 unigram 和 bigram 信息(约 1 亿条左右)放入 voldemort 后，以单独服务的形式提供访问(java client 模拟用户查询访问)，切分速度是每秒大约 2 507 个。而目前的 state-of-art(切分速度最快的)，即文献[7]的方法，理论最快的速度是 3 000 个/秒，但其测试的前提是将所有算法需要的 n-gram 信息以及待测试的查询，都放入单机内存中，这显然是不现实的。事实上，在

采用与本文类似的测试场景后，其达到的切分速度是每秒大约400个。显然，这与本文方法的切分速度依然有较大的差距。

总体上，我们的方法在保证简单高效性的基础上，依然能取得不错的可比较的切分效果（在查询精确性指标上差距在3%以内），显然是一种更为实用的查询分析方法。

4.3 查询结构分析对检索性能的影响

如同本文之前提到的，查询结构分析对外的表现是如何对用户查询进行有效的切分。而查询结构分析的目的是：识别出某一子查询为短语查询，进而指导后续的检索打分过程。传统的关于查询切分的研究仅仅关注并止步于查询的切分精度，而没有进一步地研究查询结构的分析对实际检索性能的影响。本章节的实验是进一步地验证用户查询结构的分析，对实际检索效果的提升是否有积极作用。

我们采用Lucene作为索引和检索工具。对于一般的term查询，调用Lucene默认的打分机制；对于短语查询，我们会在Lucene默认打分的基础上，考虑词项间临近度（term proximity）的影响[1-2,6]。这里我们采用与文献[1]类似的方式，计算基于词间临近度的得分，最后对二者进行线性加权。

实验采用TREC WT10g数据集，并采用topic.451～500进行测试。对于每一个查询都实验了三种不同的查询方式：(1)根据本文的切分方法得到的切分结果（our model），然后对其中切分出的子短语部分，融合临近度信息；(2)将整个查询作为一个整体，不做任何切分（No break），即传统的不做用户查询切分的情况下，融合临近度信息的方式[1-2,6]；(3)对每两个相邻词项间都做切分（Always break），即传统的基于单个term的词袋模型。

测试结果如表2所示。可以发现，相对于这两种通常的查询处理方式，查询切分的结合对检索效果的提升有明显的帮助。在MAP指标上，依据本文方法进行了查询切分的方式，较传统的基于单个term的方式有接近55%的性能提升，较传统的不做查询结构分析，“大一统”的临近度融合方式，也有近34%的性能提升。在p@5评价指标上，更是取得了达到67%的性能提升。在p@10上，相对于这两种方式，也都取得了超过25%的性能提升。

表2 不同查询结构分析下的检索性能对比

	Our model	No break	Always break
MAP	0.251 5	0.187 5	0.162 5
P@5	0.250 0	0.150 0	0.150 0
P@10	0.250 0	0.175 0	0.200 0

通常情况下，一个用户查询会包含多个语义单元。查询结构分析能够将用户查询切分成若干“片段”，每个片段则对应一个语义单元成分。例如，给定一个查询为：“second hand car for sale in California”，本文的方法会将其切分成：[second hand car] [for sale] [in California]，对用户查询的各个语义单元进行了有效的把握。对于“No break”的查询方式，其要求查询的所有term都出现在一个文本段落或者相关文档中，这种要求对于长用户查询过于严格，从而会损失很多相关结果。而“Always break”的方式，会破坏一些完整语义单元的信息，比如[second hand car]，单独的second，hand，car，或者second hand，hand car之类的切分都不能正确的表达用户语义单元的信息，从而造成许多错误结果文档的返回。

另外，需要注意本文的方法是特别针对英文查询结构提出的，对于中文查询结构的分析并不能直接适用。原因在于英语的分词和汉语的分词是不相同的。英语的分词可以直接通过空格来划分，从而可以直接采用本文提出的逐点互信息的方法。对于中文查询的结构分析，可以建立在索引词表的基础上，以词表中的单元为粒度，然后采用与本文类似的线下并行训练，线上实时计算的方法，对用户的查询结构进行有效的分析。例如，给定查询：“人民英雄的高尚情操”，针对词表中的单元“人民”，“英雄”，采用本文类似的方法，计算彼此间的互信息，从而可以得到：[人民英雄][高尚][情操]等类似的查询切分，指导后续的检索过程。

5 总结和未来工作

查询结构分析对理解用户的查询意图，提升检索的效果有很大的帮助。本文提出了一套简单高效的基于逐点互信息的查询结构分析方法。其中离线训练部分利用了时下流行的MapReduce处理框架，能高效地处理大数据量的训练语料。同时利用了Voldemort的高效随机读性能，保证线上算法的执行。然后结合一种自下而上的在线查询树构造算法，对用户查询结构进行实时有效的分析。最后，通过在TREC WT10g上的实验，进一步证明了用户

查询的切分对检索效果的提升有很好的促进作用，这也是传统关于查询切分的研究未能关注的地方。另外，通过 4.3 节中的实验，我们发现，结合了查询切分的融合临近度的打分方式，要好于以前不考虑用户查询切分的融合临近度的方式，这也给关于词项临近度的研究工作提供了新的思路。在未来工作中，我们会在基于词项临近度的排序模型上做较为深入的研究，并尝试将其与现有的查询切分工作进行更为有效的结合，最终更好地促进检索效果的提升。

参考文献

[1] T. Tao, C. Zhai. An exploration of proximity measures in information retrieval[C]//Proceedings of SIGIR'07: 295-302.

[2] J. Bai, Y. Chang, H. Cui, et al. Investigation of partial query proximity in web search[C]//Proceedings of 17th International Conference on World Wide Web, 2008:1183-1184.

[3] Huang J., Gao J., Miao J., et al. Exploring web scale language models for search query processing [C]//Proceedings of WWW 2010.

[4] R. Jones, B. Rey, O. Madani, et al. Generating query substitutions[C]//Proceedings of 15th World Wide Web, 2006: 387-396.

[5] G. Kumaran, V. R. Carvalho. Reducing long queries using query quality predictors[C]//Proceedings of SIGIR'09, 2009: 564-571.

[6] D. Metzler, W. B. Croft. A markov random field model for term dependencies[C]//Proceedings of SIGIR'05, 2005: 472-479.

[7] K. M. Risvik, T. Mikolajewski, P. Boros. Query segmentation for Web search[C]//Proceedings of WWW 2003.

[8] S. Bergsma, Q. I. Wang. Learning noun phrase query segmentation[C]//Proceedings of EMNLP-CoNLL 2007: 819-826.

[9] B. Tan, F. Peng. Unsupervised query segmentation using generative language models and Wikipedia[C]//Proceedings of WWW 2008: 347-356.

[10] M. Hagen, M. Potthast, B. Stein, et al. The power of naive query segmentation[C]//Proceedings of SIGIR '10, 2010: 797-798.

[11] Yanen Li, Bo-June (Paul) Hsu, ChengXiang Zhai, et al. Unsupervised Query Segmentation Using Clickthrough for Information Retrieval[C]//Proceedings of SIGIR'11, 2011: 285-294.

[12] G. Mishne, M. de Rijke. Boosting web retrieval through query operations[C]//Proceedings of ECIR, 2005: 502-516.

[13] 王思力，王斌. 基于双字耦合度的中文分词交叉歧义处理方法[J]. 中文信息学报，2007，21(5)：14-18.

（上接第 6 页）

[10] S. Asur, B. A. Huberman. Predicting the Future with Social Media[J]. Arxiv preprint arXiv,2010.

[11] T. Sakaki, M. Okazaki, Y. Matsuo. Earthquake Shakes Twitter Users: Real-time Event Detection by Social Sensors[C]//Proceedings of WWW 2010: the International World Wide Web Conference, 2010: 851-860.

[12] B. Boser, I. Guyon, V. N. Vapnik. A training algorithm for optimal margin classifiers[C]//Proceedings of Fifth AnnualWorkshop on Computational Learning Theory, ACM Press: San Mateo, CA, 1992: 144-152.

[13] A. Genkin, D. Lewis, D. Madigan. Large-scale bayesian logistic regression for text categorization[J]. Technometrics. 2007,49(3):291-304.

[14] J. R. Quinlan, R. M. Cameron-Jones. FOIL: A midterm report[C]//Proceedings of 1993 European Conf. Machine Learning, Vienna, Austria, 1993: 3-20.

[15] X. Yin, J. Han. CPAR: Classification based on predictive association rules[C]//Proceedings of 2003 SIAM Int. Conf. Data Mining (SDM'03), San Francisco, CA. 2003: 331-335.

[16] C. Grier, K. Thomas, V. Paxson, et al. The underground in 140 characters or less[C]//Proceedings of ACM CCS'10, Chicago, IL, Oct. 2010.