面向CQL的语料库检索引擎的高效实现

刘廷超[1]，**鲁鹿鸣**[1,2]，**杨麟儿**[1,2*]，**王雨**[1]
[1]北京语言大学 信息科学学院
[2]北京语言大学 国家语言资源监测与研究平面媒体中心
liutingchao@hotmail.com

摘要

语料库检索工具在语言学研究领域具有举足轻重的地位，对于高效获取信息至关重要。然而，当前国内语料库检索工具在语料库检索语言上缺乏统一标准，尤其支持语料库查询语言（CQL）的中文语料库检索工具相对稀缺。在使用不同分词粒度的语料库工具进行中文语料库检索时，会遇到噪声或数据召回难问题。为应对这些挑战，我们研发了支持多粒度分词的CQL 解析器系统CAMELS：一款支持CQL 语句检索，且兼容多粒度分词，支持非词典词检索的语料库检索引擎。经过多种分词器的测试，该引擎展现出了优异的召回率，并在性能上超越了BlackLab的检索速度，为语言学工作者提供了更加易用、精准的检索工具。

关键词： 语料库查询语言 ； 语料库检索 ； 倒排索引

Efficient Implementation of a CQL-oriented Corpus Retrieval Engine

Tingchao Liu[1], **Luming Lu**[1,2], **Liner Yang**[1,2*], **Yu Wang**[1]
[1]School of Information Science, Beijing Language and Culture University
[2]National Language Resources Monitoring and Research Center
Print Media Language Branch, Beijing Language and Culture University
liutingchao@hotmail.com

Abstract

Corpus retrieval tools play an important role in the field of linguistic research and are crucial for efficient access to information. However, the current domestic corpus retrieval tools lack a unified standard in terms of retrieval language, especially the relative scarcity of Chinese corpus retrieval tools that support Corpus Query Language (CQL). When using corpus tools with different granularity cuts for Chinese corpus retrieval, the noise problem or data recall difficulty problem is encountered. To cope with these challenges, we have developed CAMELS (CQL Analyzer with Multi-lEvel Lexical System). CAMELS supports CQL statement retrieval, is compatible with multi-granularity segmentation, and supports retrieval of words that are not in the vocabulary. After testing with multiple tokenizers, the engine demonstrates excellent recall and outperforms BlackLab's retrieval speed in performance, providing linguists with a more easy-to-use and precise retrieval tool.

Keywords: corpus query language , corpus retrieval , inverted index

* 通讯作者
基金项目：教育部人文社科青年基金（23YJCZH264）；中央高校基本科研业务费（北京语言大学梧桐创新平台，21PT04）

1 引言

随着语言学的深入研究和信息技术的发展，语料库作为语言学研究的基础资源，其重要性日益凸显。语料库检索引擎作为从这一宝贵资源中获取信息的核心工具，对于高效、准确地挖掘语言数据具有关键作用(程学旗et al., 2014)。通过语料库检索引擎，研究者可以迅速地检索到大量真实的语言使用实例，从而对语言的规律性、变异性、历时演变等方面进行深入分析。此外，语料库查询系统还能够辅助语言教学(Boulton, 2017)，为学生提供真实的语言学习材料，帮助其更好地理解和掌握语言知识。在翻译领域，翻译者可以通过对比双语语料库，寻找合适的翻译对等词，提高翻译的准确性和流畅性(Hu et al., 2019; Wieting and Gimpel, 2017)。而在自然语言处理领域，大规模的语料库和高效的查询系统则是训练语言模型、开发智能应用的基础(奚雪峰and 周国栋, 2016)，并可以推进多种NLP下游任务的研究(Chong et al., 2023; Kong et al., 2022)。然而，当前国内语料库检索引擎在技术和应用层面仍存在诸多挑战。

一方面，在各个语料库引擎之间存在检索语言不能通用的问题。现有的一些成功的语料库检索系统，其检索方法往往各不相同，如北京大学的CCL现代汉语语料库（Center for Chinese Linguistics Peking University，CCL）(詹卫东et al., 2019)、北京语言大学的BCC语料库（Beijing Language and Culture University Corpus Center，BCC）(荀恩东et al., 2016)、国家语言资源动态流通语料库（Dynamic Circulation Corpus，DCC）(朱君辉et al., 2022)，他们都拥有丰富的语料资源和多样的检索功能，但各自使用的检索语言差异显著。在检索语句方面，CCL的查询表达式由操作符、基本项、简单项、复杂项、过滤项、子句等构成，定义了13个特殊符号，能够实现相对复杂的检索表达式。BCC具备两套查询表达式，其一为针对序列语料的交互式查询检索式，由查询对象、限制条件和功能操作组成；其二为针对复杂知识挖掘检索需求设计的脚本式查询表达式，同样由三部分组成，查询逻辑通过分解成一系列函数来完成(荀恩东, 2023)。DCC 在检索语言设计上采用了不同的策略，其支持普通检索和模式检索，其中普通检索模式采用了基于语料库查询语言（Corpus Query Language，CQL）的扩展作为检索语言(Valenzuela-Escárcega et al., 2020)。尽管检索表达式的多样性和复杂性体现了语料库检索工具强大的功能，但同时也导致了语料库检索语言的非标准化问题。这意味着用户在使用不同的语料库时，需要学习并掌握各自独特的检索语法和规则。近年来，许多国内外语料库都添加了对CQL 的支持(张永伟and 吴冰欣, 2023)，例如CQPweb(Hardie, 2012)、Sketch Engine(Kilgarriff et al., 2008)、BlackLab (de Does et al., 2017) 等。这是因为CQL语法规则简单，构成的检索式可读性强，例如检索包含“学习”的句子的检索式为：`[word='学习']`，检索包含“学习”且“学习”为动词的句子的检索式为：`[word='学习' & pos='VV']`。

另一方面，语料预处理时是否对语料执行分词会导致不同语料库引擎存在数据噪音或数据召回难问题。例如，在CCL 语料库中，绝大部分查询基于未进行分词和词性标注的纯文本(张永伟and 吴冰欣, 2023)，无法直接用词性检索。CCL 所提供的模式查询功能可以用于限定词语间的搭配模式，然而这种设计会导致检索结果中存在噪音。例如，使用CCL查询语句“爱(X)不(X)”进行语料库查询时，引擎可能会返回如“恩爱得不得了”之类的结果，这是所有基于未分词语料的语料库引擎面临的共性问题，即无法保证用户输入的检索词会被匹配到一个完整的词语，查询结果中存在一定比例的噪音数据。除此之外，在使用分词处理后的语料建立语料库时，由于中文词语的切分粒度不固定且存在一定的歧义性，当用户输入的检索词与分词词表中的词语不一致时，检索系统就无法找到对应的倒排索引信息，从而无法获得期望的结果，数据召回问题由此产生。例如，对于词语“和平共处五项原则”，其在Stanford CoreNLP (Manning et al., 2014)中会被切分成为“和平”“共处”“五”“项”“原则”等词语，而在Jieba[1] 工具中被视为“和平共处”“五项原则”两个词语。THULAC[2]则会把整个“和平共处五项原则”视作一个词语。如果以词语“和平共处”为检索词，在以上述三种分词方式处理的语料构建的语料库中查询，引擎将不会返回任何结果。

针对语料库领域的现存问题，我们开发了一种新的语料库检索引擎——骆驼：**多粒度分词的CQL 解析器系统(CQL Analyzer with Multi-lEvel Lexical System, CAMELS)**[3]。该引擎的主要贡献如下：

(1) 支持使用通用CQL 语法检索。鉴于CQL在语料库领域的广泛应用，本文选择CQL 作为

[1]https://github.com/fxsjy/jieba
[2]http://thulac.thunlp.org/demo
[3]https://github.com/paineliu/camels

检索语言，以提升检索语言的通用性。我们基于ANTLR 4 (Parr, 2013) 工具实现了CQL 的编译器，将CQL 语句解析为语法树，进而将语法树转换成包含检索单元、检索范围及约束条件的检索式。这使得语料库检索引擎能够直接利用这些检索式执行语料查询。

(2) 改进了传统倒排索引结构，确保跨分词粒度查询准确性与高效性。在CAMELS引擎中，我们以分词单元为基础，对所有以某个词为起始的语料数据子串进行排序并建立索引。通过让所有索引项直接指向语料数据的方法，避免了额外增加存储空间。同时，CAMELS 引擎基于分组排序合并策略构建整体有序的语料数据索引表。在该索引表中，存储每个词语在语料数据表中的位置。根据词汇表构建的词典TRIE 树保存着索引表中对应词语的起始和结束位置。在用户检索时，引擎基于所输入的检索词的最大前缀在TRIE树中匹配范围，在有序数据索引表中查找，实现了对检索词的匹配文档的快速定位。通过对倒排索引结构的这些优化措施，CAMELS 引擎有效地解决了由分词粒度引起的语料库检索问题。

(3) 基于两阶段检索策略，提升了语料查询效率。在CAMELS引擎中，采用两阶段检索策略。在第一阶段中，引擎根据检索式中的检索范围在倒排索引表上筛选出符合条件的语料；在第二阶段中，引擎则基于给定的约束条件对上一阶段中给出的语料进行验证和进一步筛选。

我们将CAMELS 引擎与其它CQL 语料库检索引擎进行了对比，实验验证了CAMELS 引擎的高效性及对多粒度分词查询的兼容性。我们运用了多种分词器对语料进行分词与词性标注，基于此构建语料库，并使用CAMELS 引擎执行语料检索任务。我们对比了CAMELS 引擎和使用CQL 的开源语料库检索引擎BlackLab。实验结果表明在面对检索词未被包含于语料库分词词表中的情况下，BlackLab无法有效返回候选结果，而CAMELS 引擎能稳定地提供符合预期的候选项。测试结果充分证明了我们的检索引擎在召回率方面有出色表现。

2 背景

随着信息技术的发展，语料库查询系统日趋成熟，语料库检索工具也逐渐发展和完善。国内外已有一些语料库广泛地实践了检索工具的可用性。语料库可以按照其分词粒度划分为两类：词级别存储语料库（如DCC 等）和字级别存储的语料库（如CCL等）。然而，由于中文词语边界的模糊性和不确定性，使得用户在中文语料库上进行检索时，会面临有效检索词选择困难的问题。对于词级别存储语料库，用户在不清楚具体词表的情况下可能因为与系统分词不一致而导致查询不到结果；对于字级别存储语料库，用户则难以屏蔽未分词语料带来的检索噪音。这一问题不仅影响了检索的准确性和效率，也制约了现有语料库在中文语料库研究中的进一步应用。因此，如何克服中文词语边界的不确定性，提高中文语料库检索性能，成为当前亟待解决的重要问题。

2.1 语料库查询语言

语料库作为语言学研究的重要工具，其查询语言也经历了不断的演变与完善。在现有的语料库查询语言中，CQL 具有重要的地位。CQL 是从语料库查询处理器（Corpus Query Processor，CQP）发展而来的一系列语言，提供了词汇、短语和语法等语言学层面丰富检索功能(吴良平, 2023)，逐渐成为语料库检索领域的主流语言。语料库查询语言(Corpus Query Lingua Franca，CQLF)已经成为国际标准ISO 24631-1 (Bański et al., 2016)，在国际范围内得到广泛认可与应用。CQL语句示例如表1所示。

CQL 语法由检索单元、检索范围、约束条件三个部分组成(Lu et al., 2024)：

(1) **检索单元**。检索单元实现对一个词语的检索，它是构成检索式的基本组成部分。通过具体检索表达式，我们可以产生针对特定属性的查询结果。通过在指代单个词语的中括号中添加“属性名=属性值”对，检索单元可以针对某项属性检索特定词语。例如，`[word='学习']` 就表示查询所有包含“学习”这个词的语料。检索单元内的限定条件允许以逻辑运算符连接以表达对多个属性的综合检索。例如，表达式`[word='学习' & pos='VV']`查询包含“学习”作为动词的语料。检索式中的属性值也允许使用正则表达式，这使得用户能够进行更为灵活的模糊查询。

连续排列的检索单元可以表达检索空间上连续的多个词语，从而满足诸如连续短语等更为复杂的查询需求。未添加属性限制的中括号对`[]`则表示任意的词语。通过在检索单元后添加正则表达式量词可以实现非连续型短语或非连续且位置发生变异的短语(吴良平, 2023)。

(2) **检索范围**。CQL 允许使用within 关键字以限制语料检索的范围。在within 关键字前的部分由检索单元指定，在其后接续XML 标签或能够指代更大范围的检索单元序列来指定检

索式的匹配的文档数据范围。XML 标签的内容取决于语料中的定义。在默认情况下，可以使用<s/>表示将检索结果限定在一个句子内。检索范围语法可以和其它查询语句同时使用。

(3) **约束条件**。在约束条件语法中，用户以类似变量声明的形式为检索单元赋予名称，然后通过在检索式末尾添加"::"及后面的约束条件的方式，以此指定检索单元之间的逻辑关系。例如，在CQL语句："`A:[][]B:[] :: A.pos = B.pos`"中，分别为第一个和第三个检索单元定义了名称A和B，并且添加了第一单元和第三单元的词性必须相同的约束条件。

功能	示例	说明
检索单元	[word='学习']	包含"学习"的语料
	[word='编辑' & pos='VV']	包含"编辑"作为动词的语料
	[word='热爱'][word='学习']	"热爱""学习"连续出现的语料
	[word='不但'][]{1,5}[word='而且']	"不但"和"而且"间隔5字内的语料
检索范围	[word='科技']within <s / >	在同一句中包含"科技"的语料
约束条件	A:[] [] B:[][] :: A.word = B.word	包含形式如"有步骤有计划"语料

表 1: CQL语句示例

由于CQL在语法简洁性、直观性、灵活性以及扩展性等方面的优势，使用CQL作为语料库检索的主要语言，不仅能够降低学习成本，提高检索效率，还能够更好地满足用户的语料库检索需求。

2.2 倒排索引

倒排索引作为一种高效的信息检索手段，在现代信息处理和搜索引擎中发挥着关键作用(Babenko and Lempitsky, 2014; 刘兴宇, 2004)。它的主要功能是记录词项在文档中的出现情况，建立词项反向映射到文档的索引关系(Schütze et al., 2008)，从而提高搜索效率和匹配度。

在语料库设计中，倒排索引的构建是关键环节。语料库系统会根据所采用的词语、词性等标签，建立这些标签与文档之间的倒排索引信息，从而通过标签信息快速检索到含有这些标签的文档。倒排索引由两个部分组成：词项词典和倒排列表。词项词典由文档集合中出现的所有词项组成。文档ID用来唯一标识一个文档。每个词项对应一个或多个由文档ID以及词频、位置等信息组成的倒排列表，记录了该词项出现过的文档、以及在文档中的出现频率和出现位置等信息。这种索引方式的优点在于其高效性和灵活性，使得系统能迅速响应大量查询请求。具体来说，构建倒排索引包括分词、词性标注及建立索引表等步骤，随后系统会根据这些信息创建倒排索引表，记录标签与文档位置的对应关系。在检索时，用户的查询请求被解析为关键词，系统再利用已建立的倒排索引迅速查找到这些关键词对应的文档ID，从而快速定位并返回相关文档内容。

倒排索引在全文搜索领域应用广泛(Wu et al., 2013; Lin, 2009; Fontoura et al., 2006)，Lucene(Białecki et al., 2012)[1]是全文搜索领域的杰出代表，它是一套用于全文检索的高性能、可伸缩的信息检索工具包。Lucene以其强大的文本分析、索引构建和查询处理能力，为开发者提供了灵活且高效的搜索解决方案。Lucene的核心在于其倒排索引技术，能够快速地定位到包含特定关键词的文档(Gospodnetic et al., 2010)。通过分词、停用词过滤、词干提取等步骤，Lucene能够将原始文本转化为适合搜索的索引结构。Lucene在全文搜索领域有着举足轻重的地位。尽管Lucene 检索功能强大，但并不能支撑语料库检索的词语间距离、词性约束、以及语法约束等复杂检索条件的需求，无法直接用于语料库检索。然而基于Lucene优秀的性能，仍有许多语料库检索工具基于Lucene引擎构建(Elasticsearch, 2018; Shahi and Shahi, 2015)。

2.3 BlackLab

BlackLab[2] 是基于Lucene 引擎开发的一款专为语言学家设计，同时适用于历时研究和知识提取等领域的文本检索和分析工具。作为一款开源软件，BlackLab 基于Lucene 实现了强大的检索功能和灵活的使用方式。它能够索引标注文本，支持复杂模式的快速搜索，并可通过配置

[1]https://lucene.apache.org
[2]https://github.com/INL/BlackLab

实现个性化索引。此外，BlackLab 还提供了丰富的结果处理功能，如结果集分组与排序、检索单元突出显示等，使得用户能够更高效地分析和利用搜索结果。BlackLab 采用CQL 作为其语料库查询语言。

BlackLab 要求输入语料必须以词级别存储，这导致了在BlackLab 中使用CQL 检索时的分词粒度选择困惑问题。在构建中文语料库时，需要对语料数据进行分词处理。而程序的分词结果往往和人类认知并不一致。这种差异也导致用户在语料库检索时，难以确定一段文字在语料库中的具体切分形式。这使得用户在选择检索词时感到困惑(孙茂松, 1999; 詹卫东et al., 2019)。这种检索词的不确定性给使用者带来了极大的不便，不仅影响了检索的效率和准确性，还可能导致遗漏重要的信息。由于BlackLab 严格以词语边界作为检索条件，因此在使用BlackLab进行中文语料库检索时，用户只有在选择与语料数据切分相一致的检索词的情况下，才可能获得正确的检索结果。中文词汇的丰富性和切分单元的不确定性导致分词工具之间出现显著差异，使得如何选择语料库检索词问题更为突显(张文静et al., 2019)。表2以句子“在和平共处五项原则的基础上”为例，展示了Stanford CoreNLP、THULAC（THU Lexical Analyzer for Chinese）[1]和Jieba 等分词工具的结果，反映出中文词语的边界存在着模糊性。

分词工具	分词结果								
Stanford	在	和平	共处	五	项	原则	的	基础	上
THULAC	在	和平共处五项原则					的	基础上	
Jieba	在	和平共处		五项原则			的	基础	上

表 2: 分词工具结果对比。常用的中文分词工具也会因为训练语料与分词策略的不同，对同一句子产生不同的分词结果。可以看到分词一致性比较低，Stanford CoreNLP 工具包更倾向于切分出较小的单元，清华大学THULAC的分词粒度较大，而Jieba 分词工具的分词粒度适中。

3 我们的工作

我们选用CQL作为语料库的检索语言，基于C++、Python开发了高效的语料库检索引擎系统CAMELS。我们借助ANTLR 4 实现对CQL的高效精确解析；对传统的倒排索引策略进行了改进，以解决非词典词汇的匹配难题；还优化了存储结构和检索性能，通过实施两阶段检索策略，显著提升了检索速度。表3对比了CAMELS和传统语料库的异同。语料库的索引生成与检索流程如图1所示。

划分方法	噪音数据	存储空间	非词典词查询	有效查询词
基于字	高	高	✔	五项原则/五项原/项原则/五项/项原/原则/五/项/原/则/
基于词	**无**	**低**	✗	五项原则
我们的方法	低	**低**	✔	五项原则/五项原/五项/原则/五/原

表 3: CAMELS与传统语料库基本单位划分粒度的对比图。以“五项原则”为例，有效查询词是指能够召回“五项原则”所在句的查询词。以字为单位划分语料会产生意外的有效检索词，而以词为单位划分则会导致过于严格的查询词限制。CAMELS较好地平衡了两种方式的优点，并在其它方面有良好表现。

3.1 CQL语法解析

我们基于ANTLR 4 实现了CQL 解析器，该解析器接收CQL 作为输入，分析其语义，生成语法树，将语法树转换为CAMELS 引擎支持的结构化检索表达式。ANTLR 4（ANother Tool for Language Recognition 4）[2] 是一款功能强大的开源语法分析工具，允许开发者使用扩展巴斯科范式（Extended Backus–Naur Form，EBNF）来描述语法(Parr et al., 2014)。我们

[1]https://github.com/thunlp/THULAC-Python
[2]https://www.antlr.org

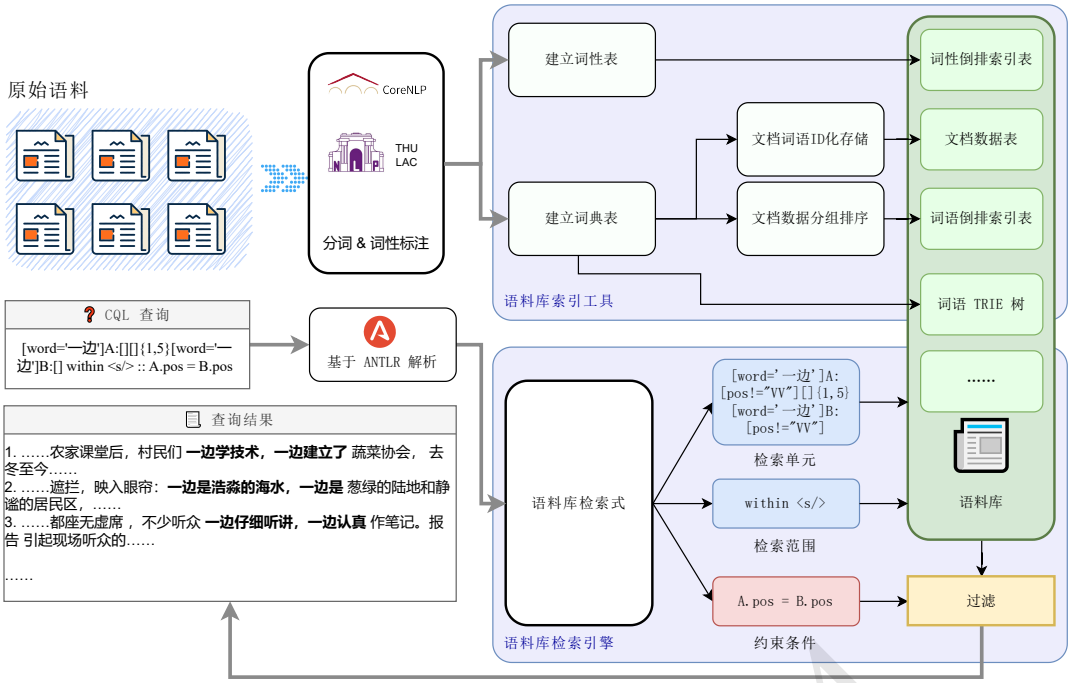

图 1: 语料库索引生成与检索流程图。输入语料被CAMELS 索引工具处理后转换为多个数据表。输入的CQL查询则被转换为CAMELS 内部检索表达式以查询相应语料。图中用CQL 的语句片段表达拆分后的检索表达式的含义，在实际引擎中，查询语句的语义被以另外的形式表述并执行。

在Corpus Query Language Parser开源项目[1] 提供的CQL语言描述文件的基础上，使用ANTLR 4生成了CQL的语法解析器，并基于ANTLR Listener 遍历语法树以实现语义分析和结构化查询表达。分析器构建的一个语法树样例如图2所示。

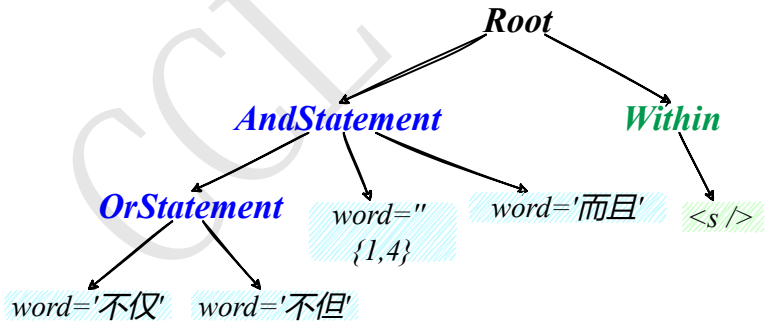

图 2: CQL语法树示意图。以CQL 语句“[word=‘不仅’ |word=‘不但’] []{1,4} [word=‘而且’] within <s/ >”为例，ANTLR生成的语法树结构将该输入语句按层次解析。在这之后，我们基于ANTLR Listener 遍历语法树并完成语义分析。

CQL 经由解析器构建出CAMELS 检索表达式的三个核心组成部分：查询语句、约束条件以及检索范围。查询语句是构成检索表达式的基础元素，它们对应于CQL查询语句中的检索单元。这些查询语句是可以执行的具体查询指令，能够直接对语料库进行数据检索。范围约束由CQL中的检索范围所指定的，其限定了检索单元在语料库中的匹配范围，用户可以指定的范围是句子范围、段落范围或者文档范围。约束条件是由CQL查询语句中的检索条件转换而来的，在检索表达式中起着至关重要的作用。约束条件确保了检索单元之间的关系符合用户的查

[1]https://github.com/exquery/corpusql-parser

询意图，它们对检索单元进行逻辑上的连接和限制，从而保证了检索结果满足逻辑关系。由解析器构建出的检索表达式可以被CAMELS 引擎直接运行。

3.2 倒排索引表设计

语料库索引工作主要是创建倒排索引表。通常建立语料库索引需要以下步骤：原始文本经过统一编码、段落分割、句子分割等预处理流程后，形成统一格式的语料数据；之后使用语料标注工具或人工标注进行分词和必要信息标注，最后通过语料库索引工具创建倒排索引表以及其它数据表(Kaushik et al., 2004)。由于不同的检索标签的检索需求存在差异，因此需要分别设计倒排索引策略。对于词性检索来说，其检索词是固定且词性体系，因此对于词性采用传统的倒排索引结构设计，直接建立词性到文档之间的索引。

对于词语检索来说，由于中文词语边界的模糊性，需要解决非语料库词表词语的检索匹配问题，因而无法使用传统的倒排索引方式建立索引。词语检索需要具有跨词语边界的检索能力，即对词语倒排索引表的访问需要打破词语边界。例如，对于表2中的例子，用户可能会将“和平共处”（在语料库中是一个完整词语）拆分为“和平”“共处”进行搜索，或者将“五项”（在语料库中是分开的“五”和“项”）组合成一个词进行搜索。此外，在建立倒排索引时，还需要根据语料库提供的范围约束条件，建立文档、段落、句子等多级倒排索引结构。

为实现跨词语边界的检索能力，我们设计了四步流程来构建词语倒排索引表：（1）进行文档词语的ID化转换。我们使用多种分词工具对语料进行分词，保留多种粒度切分，生成词典表，进而将文档内容转换为词语ID序列。（2）对文档数据进行分组排序。文档转换为词语ID序列后，遍历句子中的每个词语，并以此词语为起始点，后续第一个非中文字符为结束点，确定一个子句，然后将这个子句加入到待排序子句列表中。此列表仅记录子句在全部文档中的偏移位置，并不产生新的句子实例。由于待排序子句的数量众多，我们采用句首字符进行分组排序，以提高处理效率。（3）创建词语倒排索引表。当待排序子句完成排序后，会生成一张从语料库中任意词开始的子句排序列表。这使得我们可以利用二分查找法在这张表上进行字符串的匹配范围查询。CAMELS 引擎将子句作为整体字符串与检索词进行比较，而非以词语为单位，从而实现了即便检索词不在词表中也能进行有效检索。最终，我们将子句排序索引以倒排索引表的形式存储，此表实现了从子句到文档的映射。（4）生成词语TRIE树，建立词语到倒排索引表的映射关系。TRIE 树以每个字符为节点进行构建，每个节点下都存储了当前节点在倒排索引表中的匹配范围。在查询关键词的匹配范围时，如果关键词与TRIE树的节点完全匹配，则直接使用匹配节点的检索范围；如果未完全匹配，我们则在TRIE树与检索词的最长匹配节点的范围内进行二分查找，以确定搜索关键词的精确匹配范围。

检索过程的示例如图3所示。当用户搜索“和平共处五项原则”时，检索引擎首先在词典TRIE树上进行匹配，找到最长匹配节点“和平”，并获取其在倒排索引表中的检索范围。由于检索词并未完全匹配，CAMELS引擎需在该范围内进行二分查找，以进一步确定“和平共处”的检索范围，进而将其范围内的候选项与约束条件进行匹配过滤，以获得最终的检索结果。

3.3 两阶段检索策略

我们采用两阶段检索策略以保证查询效率与精确性。语料库检索不仅需要实现检索单元的精确匹配，还需验证各检索单元在语料中的位置和距离关系是否符合预设约束。在处理多段检索单元时，先筛选出符合匹配条件的语料，再进行约束条件的验证，可以显著提升检索效率。在CAMELS 检索过程中，我们设置如下两个阶段：

（1）执行检索单元的匹配。在本阶段，引擎执行忽略位置和距离约束的检索表达式，从语料库中检索出所有可能的匹配语料。对于简单检索式，直接查询对应的倒排索引表，返回匹配结果。对于包含逻辑运算的复合检索式，引擎首先执行复合检索式中的简单检索式并取得匹配结果，之后再根据简单检索式之间的逻辑运算关系进行运算，以获得复合检索式的匹配结果。

（2）根据约束条件对第一阶段获取的语料进行筛选。具体而言，引擎依据位置和距离等约束条件，对上一阶段返回的匹配文档进行约束匹配，返回相应的详细信息。引擎从文档的开始位置，逐一核查检索单元的匹配位置是否符合预设的约束条件。由于一个文档内可能存在多个与检索单元相匹配的位置，因此采用动态规划算法，从起始检索单元开始，逐个检查其匹配位置是否满足过滤条件。只有当所有检索单元都符合要求时，我们才将该文档添加到最终的候选列表中。

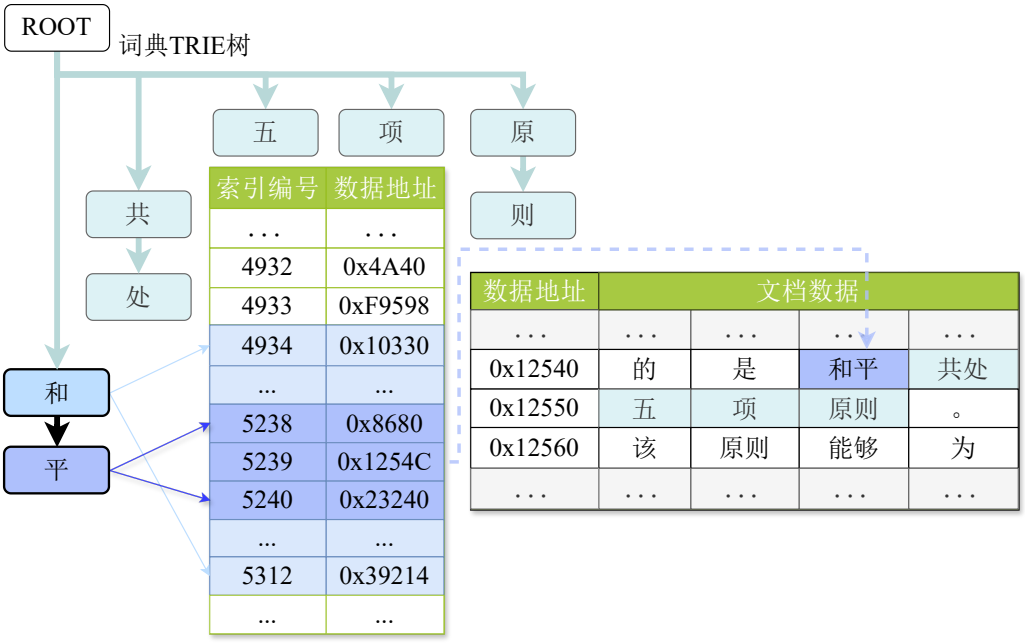

图 3: 倒排索引检索示意图

值得注意的是，合并不同检索单元以及复合检索式内部的简单检索结果时，传统的倒排索引表存储有序文档ID列表，可以通过跳表操作快速合并列表；然而，在我们的方法中，由于词语不是固定的索引单位，且对应的文档ID无序，因此无法采用传统的合并方式。为解决这个问题，我们采用了标记位匹配法，建立一个与文档数量相同的位数组，其中每位表示对应文档是否与检索词匹配。在合并两个检索表达式的结果时，只需对匹配标记进行逻辑运算，即可迅速完成合并操作。

4 实验

在本章，我们详细介绍了实验使用的数据与方法。在4.1节，我们介绍了实验使用的数据集。在4.2节，本文对比了BlackLab 与CAMELS 的检索词召回率。在4.3节，本文对比了BlackLab与CAMELS的检索速度。

4.1 数据集

实验所采用数据集为《人民日报》1946年至2022年的新闻文本语料。对语料分别使用CoreNLP以及THULAC 分词工具进行处理，再使用BlackLab 与CAMELS 引擎分别对处理过的文件建立索引，生成语料数据库。我们采用人工编撰和机器生成结合的方式编写CQL测试用例。我们基于随机抽取的语料中的句子，自动生成并人工核验了CQL 测试语句，在BlackLab以及CAMELS引擎上进行检索词召回测试和检索速度测试。

4.2 检索词召回测试

我们从《人民日报》语料库中随机抽取1000条长度在5至20个汉字的句子，并应用Jieba分词工具进行分词。构造CQL测试用例时，我们首先从句子中随机选取部分词长大于1的词语W_i，并以“`[word='`W_i`']`”格式生成测试用例。如果该用例无法在相应引擎中返回指定的来源句子，则认为召回失败。我们对于这些构造的CQL测试用例，分别在使用CoreNLP 工具和THULAC 工具建立的语料库上进行检索词的召回测试。召回测试的样例在表4中阐明。实验结果见表5。经测试，CAMELS在所有测试用例上达到了100%的召回率，实验结果表明CAMELS 引擎可以实现跨分词粒度的检索。

分词不一致性是导致召回失败的关键因素。在分析CoreNLP和THULAC分词工具在处理特定测试用例（特别是召回失败案例）中的分词行为后，可以明确识别出这种不一致性：分词工具对语料库的处理结果与检索词之间存在的显著差异。以“至关重要”为例，CoreNLP将其分

检索目标	CQL	BL-S	BL-T	CAMELS
完善农村**宅基地**管理制度。	[word='宅基'][word='地']	✔	✗	✔
	[word='宅基地']	✗	✔	✔
这说明扩大内需还要**加把劲**。	[word='加把劲']	✔	✗	✔
	[word='加'][word='把'][word='劲']	✗	✔	✔
两个因素**至关重要**。	[word='至关重要']	✗	✔	✔
	[word='至关'][word='重要']	✔	✗	✔
我只能**干着急**。	[word='干着急']	✗	✔	✔
	[word='干'][word='着急']	✔	✗	✔
这些都是**未解之谜**。	[word='未解之谜']	✗	✗	✔

表 4: 检索词召回实验的示例表。BL-S列代表使用Stanford CoreNLP分词建立的BlackLab语料库，BL-T代表使用清华THULAC分词建立的BlackLab语料库，CAMELS表示本文的语料库。✔表示该CQL在其对应语料库中可以成功召回，✗则表示召回失败。

割为“至关”和“重要”两个独立词语，而THULAC则将其视为一个完整的词语“至关重要”。在处理“未解之谜”时，CoreNLP将其分割为“未解”“之”“谜”三个独立词语，而THULAC则将其分割为“未”“解”“之”“谜”四个独立词语。这种分词差异导致了在现有检索系统中使用用户期望的的检索词检索时，可能会出现无法准确召回目标信息的问题。CAMELS凭借其多粒度分词支持和跨词语检索功能，有效解决了分词不一致性导致的数据召回问题，实现了100%的召回率，显著提升了检索的准确性和效率。

4.3 检索速度测试

检索速度测试数据包含50条人工编写的测试用例，和450条自动生成的从人民日报中随机抽取的多字词测试用例，合计500条。我们参考Lu 等(2024) 提出的CQL 生成方法编写检索速度查询测试用例。我们基于汉语增强依存(Yu et al., 2022)抽取搭配并生成检索单元，并随机生成多字词或多字词与其词性的组合的查询语句。针对每一条测试用例，分别应用BlackLab（语料基于Stanford分词）与CAMELS进行12次查询，为确保数据的准确性，我们排除了第一次查询的结果（因为BlackLab在进行第一次查询时会加载数据，导致查询速度较慢），也去除了运行时间为0的异常数据。对剩余的记录取平均值得到最终的结果。实验表明，CAMELS 引擎平均响应时间为13.57毫秒，BlackLab平均响应时间为14.73毫秒，CAMELS 引擎的检索速度相较BlackLab 有7.88% 的提升。实验结果如表5所示。

性能 / 系统	RT(ms)	BR(%)	Recall
CAMELS	**13.57**	**58.2**	**100**
BlackLab	14.73	41.8	89.6(S) / 93.4(T)

表 5: 检索速度对比表。RT指平均响应时间，BR指更优比例，表示在所有测试中取得更优结果的数量占总测试数的比例。Recall指在检索词召回实验中的召回比例。表格分别给出了用在Stanford CoreNLP工具分词语料上建立的语料库（S）和在THULAC工具分词语料上建立的语料库（T）检索的结果。CAMELS在召回率上做到了100%。

5 结论

本文深入探讨了中文语料库检索工具在检索语言通用性和非词典词匹配召回方面的问题，并针对这些问题设计研发了一款新型的语料库检索引擎。在解决检索语言通用性问题时，我们采用了国际上广泛使用的CQL语言作为语料库的检索语言，并通过ANTLR工具将CQL解析为语法树，实现了高效的检索表达式解析。在解决非词典词匹配召回问题时，我们改进了传统的倒排索引结构，构建了整体有序的语料数据索引表，优化了词语检索匹配策略，实现了非词表词语的有效匹配。同时，我们采用了二阶段检索策略和候选标记位过滤方案，显著提高了检索

速度。实验结果显示，与业界知名的BlackLab引擎相比，我们的检索引擎在非词典词检索时，仍能返回预期结果，且在检索速度方面具有明显优势。综上所述，本研究通过自研查询引擎实现了非词典词的检索，并展示了CQL的简洁性和直观性，为语料库检索技术的研究和应用提供了有益参考和高效工具。

参考文献

Artem Babenko and Victor Lempitsky. 2014. The inverted multi-index. *IEEE transactions on pattern analysis and machine intelligence*, 37(6):1247–1260.

Piotr Bański, Elena Frick, and Andreas Witt. 2016. Corpus Query Lingua Franca (CQLF). In Nicoletta Calzolari, Khalid Choukri, Thierry Declerck, Sara Goggi, Marko Grobelnik, Bente Maegaard, Joseph Mariani, Helene Mazo, Asuncion Moreno, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 2804–2809, Portorož, Slovenia, May. European Language Resources Association (ELRA).

Andrzej Białecki, Robert Muir, Grant Ingersoll, and Lucid Imagination. 2012. Apache lucene 4. In *SIGIR 2012 workshop on open source information retrieval*, page 17. sn.

Alex Boulton. 2017. Corpora in language teaching and learning. *Language Teaching*, 50(4):483–506.

Ruining Chong, Luming Lu, Liner Yang, Jinran Nie, Shuhan Zhou, Yaoxin Li, and Erhong Yang. 2023. Mcts: A multi-reference chinese text simplification dataset.

Jess de Does, Jan Niestadt, and Katrien Depuydt. 2017. Creating research environments with blacklab. *CLARIN in the Low Countries*, pages 245–257.

BV Elasticsearch. 2018. Elasticsearch. *software], version*, 6(1).

Marcus Fontoura, Ronny Lempel, Runping Qi, and Jason Zien. 2006. Inverted index support for numeric search. *Internet Mathematics*, 3(2):153–185.

Otis Gospodnetic, Erik Hatcher, and Michael McCandless. 2010. *Lucene in action*. Simon and Schuster.

Andrew Hardie. 2012. Cqpweb—combining power, flexibility and usability in a corpus analysis tool. *International journal of corpus linguistics*, 17(3):380–409.

J Edward Hu, Rachel Rudinger, Matt Post, and Benjamin Van Durme. 2019. Parabank: Monolingual bitext generation and sentential paraphrasing via lexically-constrained neural machine translation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 6521–6528.

Raghav Kaushik, Rajasekar Krishnamurthy, Jeffrey F Naughton, and Raghu Ramakrishnan. 2004. On the integration of structure indexes and inverted lists. In *Proceedings of the 2004 ACM SIGMOD international conference on Management of data*, pages 779–790.

Adam Kilgarriff, Pavel Rychly, Pavel Smrz, and David Tugwell. 2008. The sketch engine. *Practical Lexicography: a reader*, pages 297–306.

Cunliang Kong, Yun Chen, Hengyuan Zhang, Liner Yang, and Erhong Yang. 2022. Multitasking framework for unsupversied simple definition generation.

Jimmy Lin. 2009. Is searching full text more effective than searching abstracts? *BMC bioinformatics*, 10:1–15.

Luming Lu, Jiyuan An, Yujie Wang, Liner Yang, Cunliang Kong, Zhenghao Liu, Shuo Wang, Haozhe Lin, Mingwei Fang, Yaping Huang, and Erhong Yang. 2024. From text to cql: Bridging natural language and corpus search engine. *ArXiv*, abs/2402.13740.

Christopher D Manning, Mihai Surdeanu, John Bauer, Jenny Rose Finkel, Steven Bethard, and David McClosky. 2014. The stanford corenlp natural language processing toolkit. In *Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations*, pages 55–60.

Terence Parr, Sam Harwell, and Kathleen Fisher. 2014. Adaptive ll (*) parsing: the power of dynamic analysis. *ACM SIGPLAN Notices*, 49(10):579–598.

Terence Parr. 2013. The definitive antlr 4 reference. *The Definitive ANTLR 4 Reference*, pages 1–326.

Hinrich Schütze, Christopher D Manning, and Prabhakar Raghavan. 2008. *Introduction to information retrieval*, volume 39. Cambridge University Press Cambridge.

Dikshant Shahi and Dikshant Shahi. 2015. Apache solr: an introduction. *Apache Solr: A practical approach to enterprise search*, pages 1–9.

Marco A Valenzuela-Escárcega, Gus Hahn-Powell, and Dane Bell. 2020. Odinson: A fast rule-based information extraction framework. In *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pages 2183–2191.

John Wieting and Kevin Gimpel. 2017. Paranmt-50m: Pushing the limits of paraphrastic sentence embeddings with millions of machine translations. *arXiv preprint arXiv:1711.05732*.

Hao Wu, Guoliang Li, and Lizhu Zhou. 2013. Ginix: Generalized inverted index for keyword search. *Tsinghua Science and Technology*, 18(1):77–87.

Jingsi Yu, Shi Jialu, Liner Yang, Dan Xiao, and Erhong Yang. 2022. Transformation of enhanced dependencies in chinese. In *Proceedings of the 21st Chinese National Conference on Computational Linguistics*, pages 99–109.

刘兴宇. 2004. 基于倒排索引的全文检索技术研究. Ph.D. thesis, 武汉: 华中科技大学.

吴良平. 2023. Cqp语法赋能语言研究及语言学习. 语料库语言学, 10:98–114.

奚雪峰and 周国栋. 2016. 面向自然语言处理的深度学习研究. 自动化学报, 42(10):1445–1465.

孙茂松. 1999. 谈谈汉语分词语料库的一致性问题. 语言文字应用, (2):90–93.

张文静, 张惠蒙, 杨麟儿, and 荀恩东. 2019. 基于lattice-lstm的多粒度中文分词. 中文信息学报, 1:7.

张永伟and 吴冰欣. 2023. 基于网络的第四代语料库分析工具核心功能评介. 当代语言学, 25(04):611–624.

朱君辉, 刘鑫, 杨麟儿, 师佳璐, and 杨尔弘. 2022. 文心语料库检索平台的研制. 第十二届全国语言文字应用学术研讨会.

程学旗, 靳小龙, 王元卓, 郭嘉丰, 张铁赢, and 李国杰. 2014. 大数据系统和分析技术综述. 软件学报, 25(9):1889–1908.

荀恩东, 饶高琦, 肖晓悦, and 臧娇娇. 2016. 大数据背景下bcc语料库的研制. 语料库语言学, 3:93–109+118.

荀恩东. 2023. 自然语言结构计算*BCC*语料库. 自然语言结构计算BCC语料库.

詹卫东, 郭锐, 常宝宝, 谌贻荣, and 陈龙. 2019. 北京大学ccl语料库的研制. 语料库语言学, 6(01):71–86+116.