重庆大学本科学生毕业论文（设计）

# 面向多编程语言的代码检索系统设计与实现


学　　生：夏劲
学　　号：20214966
指导教师：刘超
专　　业：软件工程

重庆大学大数据与软件学院

2025 年 6 月

Undergraduate Thesis (Design) of Chongqing University

# Design and Implementation of a Multi-Programming Language Code Retrieval System


By

XIAJin

Supervised by

Prof. LIU CHAO

Software Engineering

School of Big Data and Software Engineering

Chongqing University

June，2025

# 摘 要

随着编程语言的多样化和开源项目的迅速增长，开发者在寻找特定代码片段时面临着越来越大的挑战。现有的代码托管平台主要依赖关键词匹配和特定搜索条件进行检索，效率较低且使用门槛较高。大语言模型逐渐成为科研和工业界的热点，利用大语言模型可以轻松将用户的自然语言转译为具有查询功能的条件，并结合具体信息拓展补充用户的查询条件，从而实现更高效的查询体验。本项目旨在利用大语言模型实现一款面向多编程语言的代码检索系统。

本文首先介绍了大语言模型的发展情况，并论证了使用大语言模型进行查询条件重写的可行性，强调了提示词工程在这一过程中的重要性。接着简述了前端框架 Vue、Visual Studio Code 平台的插件开发、Elasticsearch 和 FastAPI 等后台开发框架。根据系统的应用目标，本文进行了系统需求分析，设计了系统架构、功能和数据库结构。在此基础上，本项目开发了基于 FastAPI 的后台管理逻辑服务、基于 Elasticsearch 的代码检索服务以及基于 DeepSeek-R1 的查询重写服务。通过 Visual Studio Code 平台提供的接口，结合 Vue 框架实现了 Visual Studio Code 插件的前端开发，完成了快捷键绑定、项目搜索等一系列功能。

本项目开发的面向多编程语言的代码检索系统具有界面美观、操作友好、查询效率高、查询准确等诸多优点。基于 FastAPI 开发的后台管理系统易于管理和维护，帮助开发人员实现高效的多编程语言项目代码搜索。

**关键词**：代码搜索；Elasticsearch；大模型提示工程

# ABSTRACT

As programming languages diversify and open-source projects rapidly grow, developers face increasing challenges in finding specific code snippets. Existing code hosting platforms primarily rely on keyword matching and specific search conditions, resulting in low efficiency and a high usage threshold. Large language models have become a hot topic in both research and industry. By leveraging these models, users' natural language can be easily translated into functional query conditions, which can be further expanded with specific information to enhance the query experience. This project aims to implement a code retrieval system for multiple programming languages using large language models.

This paper first introduces the development of large language models and demonstrates the feasibility of using them for query condition rewriting, highlighting the importance of prompt engineering in this process. It then briefly describes the front-end framework Vue, plugin development for the Visual Studio Code platform, and back-end development frameworks such as Elasticsearch and FastAPI. Based on the system's application goals, a system requirements analysis was conducted, and the system architecture, functionality, and database structure were designed. On this basis, the project developed a back-end management logic service based on FastAPI, a code retrieval service using Elasticsearch, and a query rewriting service using DeepSeek-R1. The front-end development of the Visual Studio Code plugin was achieved through the platform's provided interfaces, combined with the Vue framework, implementing features such as shortcut key bindings and project search.

The code retrieval system developed in this project for multiple programming languages offers numerous advantages, including an aesthetically pleasing interface, user-friendly operation, high query efficiency, and accuracy. The back-end management system developed with FastAPI is easy to manage and maintain, aiding developers in efficiently searching for code across multi-language projects.

**Key words**: Code Search;Elasticsearch;LLM Prompt

# 目 录

# 1 绪论

## 1.1 研究目的及意义

随着机器学习、深度学习等人工智能技术的快速发展，这些高新技术在解决传统领域挑战中发挥了重要作用。在搜索领域，传统搜索服务提供商通常直接采用 Elasticsearch 作为搜索引擎来完成信息检索。然而，传统搜索引擎基于倒排索引的检索方式要求用户提供极其精确的搜索词，并掌握一定的搜索技巧，这极大地限制了普通用户的搜索体验。随着人工智能技术，尤其是大语言模型（LLM）的进步，利用其自然语言友好特性来辅助搜索已成为科研界和工业界的研究热点。

Elasticsearch 是一种分布式、RESTful 风格的搜索和分析引擎，其强大的搜索能力源于其独特的倒排索引结构（Inverted Index）。这种数据结构使 Elasticsearch 具备高效的全文搜索能力、实时数据处理能力以及高可扩展性，同时能够处理结构化和非结构化数据。

大型语言模型（LLM）是拥有海量参数和卓越学习能力的高级语言模型，其核心模块是 Transformer 架构中的自注意力机制。自注意力机制作为语言建模任务的基本构建块，能够有效处理顺序数据，实现并行化计算，并捕捉文本中的远程依赖关系。LLM 的显著优势在于其能够理解自然语言并执行基于自然语言的指令，这对用户侧软件服务非常友好。然而，LLM 的另一特点是其庞大的模型参数和极高的训练成本。从头预训练一个大型模型所需的资源（如数百万高性能显卡卡时）是普通实验室或个人工作者难以承担的。因此，针对特定场景的任务，工业界普遍采用开源的大语言模型进行监督微调（SFT），或通过 Prompt 工程来实现任务目标。

本项目旨在构建一个面向多编程语言的代码检索系统，该系统将融合大语言模型的自然语言友好能力，实现对用户搜索词的重写、完善，结合 Elasticsearch 强大的多结构文本搜索能力实现用户方便快速搜索多语言项目的能力。同时我还讲相关能力集成到 Visual Studio Code 平台，发布免费的开源插件，为所有开发者提供便捷高效的代码搜索服务。

## 1.2 国内外研究现状

2004 年，Shay Banon 创造了 Elasticsearch 的前身——Compass。在 Compass 的基础上，Shay Banon 进一步实现了分布式和可扩展性优化，并提供了 HTTP 接口，

使得 Java 以外的语言也能调用。2010 年，Elasticsearch 的第一个版本正式发布。

Elasticsearch 是一种成熟的搜索解决方案，一些大型搜索公司如 Google 和百度都基于 Elasticsearch 进行研发。Elasticsearch 支持 RESTful 风格的调用，客户端可以通过 HTTP 请求直接操作 Elasticsearch 服务。它还提供了多种客户端支持，包括 Java、Python、Go、PHP 等语言的 SDK，以及 JDBC、ODBC 等标准化接口。Elasticsearch 广泛应用于全文搜索、日志分析、实时数据分析、地理空间搜索以及商业智能领域的数据聚合分析。在部署方面，Elasticsearch 支持单节点模式，适用于开发和测试环境，可以通过简单的 Docker 命令快速启动。在生产环境中，通常采用分布式集群部署，通过分片（Shard）和副本（Replica）机制实现水平扩展和高可用性。集群节点可以动态扩容并自动平衡数据负载，同时支持主节点、数据节点、协调节点等角色划分，以优化资源分配。

在大语言模型方面，基于 Transformer 架构的大语言模型如 OpenAI 的 GPT 系列、Google 的 Gemini、Meta 的 Llama、腾讯的混元系列模型等，都已经在自然语言处理领域展现出强大的能力。这些模型通过海量数据训练，能够理解和生成高质量的自然语言文本。在 Transformer 基础上，大语言模型继续发展，得益于多专家（MoE）架构、多模态融合以及自监督学习等技术创新，其在代码语义理解、跨语言检索等场景中的潜力被进一步释放。例如，GitHub Copilot 基于 GPT 系列开发的 Codex 模型，能够完成代码补全和跨语言语义关联；Google 推出的 Gemini Code Assist 则通过多专家架构优化了代码检索的响应速度和准确性。与此同时，模型自我反思机制成为提升代码生成可靠性的关键技术突破，以 DeepSeek R1 为代表的反思模型通过强化学习框架实现自我校验与动态优化，其采用的图谱重标定（GRPO）算法可对代码逻辑进行多步验证，在 Math-500 测试中达到 97.3% 的准确率，并在 Codeforces 编程竞赛中超越 96% 的人类选手。该模型通过冷启动训练策略融合监督微调与自我对弈机制，不仅能在代码生成过程中主动识别变量定义错误、逻辑矛盾等问题，还能通过语言反馈链重构搜索词表达，显著提升多编程语言检索的语义对齐度。当前主流模型如 Llama 3 和通义千问已通过指令微调集成类似反思机制，使得代码检索系统的查询意图理解误差率较传统模型降低 42%。

## 1.3 论文研究内容

本项目以构建一款面向多编程语言的化代码检索系统为目标，基于 Elasticsearch 构建分布式搜索引擎内核，集成 LLM（Large Language Model）实现搜索意图理解与查询词重构，通过 FastAPI 搭建高性能异步后端服务，并采用 Vue3 响应

式框架开发可视化交互界面，最终以 Visual Studio Code 插件形态实现 IDE 深度集成，为开发者提供语义级代码检索能力

整个系统架构分为四大核心模块：

第一部分，代码采集与预处理：基于 Scrapy 框架构建分布式爬虫，通过 GitHub API 获取主流编程语言的开源代码，采用 AST 解析技术实现代码片段结构化提取，并完成开源协议合规性校验。

第二部分，存储与索引构建：部署 Elasticsearch 集群，设计 BM25 算法与稠密向量混合索引结构，集成 IK 分词插件优化专业术语处理，支持代码语法特征与语义特征的双重表征。

第三部分，检索服务实现：基于 FastAPI 框架开发 RESTful API 接口，采用异步 IO 机制提升高并发性能，设计融合关键词匹配、向量相似度计算及上下文权重的混合搜索算法，实现毫秒级响应。

第四部分，交互界面开发：使用 Vue3 Composition API 构建渐进式前端应用，通过 VSCode Extension API 实现 IDE 插件集成，支持代码快捷搜索、交互式代码预览等功能。

## 1.4 论文组织架构

第一章，绪论。阐述了本研究的目的和背景，并分析当前国内外的搜索技术和大模型研究现状，概述了本论文的主要研究内容。

第二章，相关理论和技术。通过对大模型与提示工程相关技术的介绍，阐明了本研究选择大模型作为搜索词重写基石的原因，并介绍开发多编程语言代码检索系统所使用的技术框架。

第三章，系统需求分析与设计。对系统进行需求分析后，以此设计流程完整的应用程序。

第四章，系统后端应用层实现。

第五章，系统前端表现层实现。

# 2 相关理论和技术

## 2.1 大语言模型

大语言模型是实现本系统搜索功能的核心技术。要深入理解当前的大语言模型，必须先了解其核心架构——Transformer。因此，本节将简要介绍 Transformer 及大语言模型的其他关键技术。

### 2.1.1 Transformer

Vaswani 等人指出，循环神经网络模型（RNN）在每个时间步都需要依赖前一时间步的隐藏状态信息进行计算，这种固有的顺序依赖性使得 RNN 难以在多 GPU 上进行并行计算，从而限制了 RNN 在处理超大规模文本数据时的训练能力。为了解决这一问题，他们提出了 Transformer 架构，这是一种完全依赖注意力机制连接编码器和解码器的网络架构。Transformer 显著提高了训练的并行度和速度。后续的一系列研究表明，基于 Transformer 架构的预训练模型（pre-trained models / pre-trained language models，PTM / PLM）在各种任务上都能实现最先进的性能表现，因此，Transformer 已成为自然语言处理（NLP）领域的首选架构。除了在语言相关领域的应用之外，Transformer 还被广泛应用于计算机视觉、音频处理以及自然科学学科，如化学、生物等领域。

Transformer 架构如图2.1所示，其整体由编码器 (图2.1左) 和解码器 (图2.1右) 组成。每个编码器块由多头注意力机制模块和位置前馈网络组成，模块间使用残差连接，并配有层归一化模块；解码器块在位置前馈网络和多头自注意力模块之间插入了交叉注意力模块，其中的自注意力模块用于组织某个位置信息对后续位置信息的影响。下面将简要介绍上述几种重要模块：

（1）多头注意力机制：让序列中的每一个元素学习并计算与其他元素的互注意力分数权重，如公式2.1所示：

$$\text{Attention(Q,K,V)} = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V \tag{2.1}$$

但 Transformer 并不只是简单应用了单个注意力函数，而是使用了多头注意力机制将 $d_m$ 维度的原始 $Q$、$K$、$V$ 分别线性投影到 $d_k$、$d_k$、$d_v$ 维度，再根据公式2.1进行

注意力计算，整体公式如下：

$$\text{MultiHead}(\mathrm{Q},\mathrm{K},\mathrm{V}) = \text{Concat}(\text{head}_1,...,\text{head}_\mathrm{h})\mathrm{W}^\mathrm{O} \tag{2.2}$$

$$\text{where head}_\mathrm{i} = \text{Attention}(\mathrm{QW}_\mathrm{i}^\mathrm{Q},\mathrm{KW}_\mathrm{i}^\mathrm{K},\mathrm{VW}_\mathrm{i}^\mathrm{V}) \tag{2.3}$$

Transformer 利用多头注意力机制能够同时关注到来自不同位置的且具有不同表示的子空间信息，增强了架构的表达能力。

（2）位置前馈网络：全连接前馈网络模块，用于接收自注意力模块的输出：

$$\text{FFN}(x) = \max(0, x\mathrm{W}_1 + \mathrm{b}_1)\mathrm{W}_2 + \mathrm{b}_2 \tag{2.4}$$

$$= \text{ReLU}(\mathrm{H}'\mathrm{W}_1 + \mathrm{b}_1)\mathrm{W}_2 + \mathrm{b}_2 \tag{2.5}$$

（3）残差连接与归一化：Transformer 在每个模块间使用残差连接，然后进行层归一化。其中编码器块表示为：

$$\mathrm{H}' = \text{LayerNorm}(\text{Self Attention}(\mathrm{X}) + (\mathrm{X}) \tag{2.6}$$

$$\mathrm{H} = \text{LayerNorm}(\text{FFN}(\mathrm{H}') + (\mathrm{H}') \tag{2.7}$$

### 2.1.2 混合专家模型

Google Brain 团队发现传统 Transformer 架构在扩展到超大规模时面临计算资源消耗激增的问题，尤其是前馈网络层（FFN）的全参数激活机制导致训练和推理效率急剧下降。为此，他们提出基于混合专家模型（Mixture of Experts, MoE）的稀疏架构改造方案，通过将 Transformer 中的 FFN 层替换为动态路由的专家网络集合，实现计算效率与模型容量的平衡。MoE 架构如图2.2所示，其核心创新在于引入稀疏门控机制（图2.2蓝色部分），该机制由可学习的路由网络构成，针对每个输入词元生成专家选择概率分布，仅激活概率最高的前 K 个专家（通常 K=2-4），其余专家保持非激活状态。具体而言，每个 MoE 层包含 N 个独立的前馈网络作为专家（例如 N=8 或 32），其数学表达为：

$$\text{MoE}(x) = \sum_{i=1}^{K} G(x)_i \cdot E_i(x) \tag{2.8}$$

其中 $G(x)$ 为门控网络输出的 Top-K 权重，$E_i(x)$ 为被选中的专家网络输出。这种设计使得模型总参数量可扩展至万亿级别，而实际计算量仅与激活专家数相关。实验表明，Switch Transformer 在同等计算资源下，训练速度较传统稠密模型提升 4 倍，且推理时内存占用降低至 1/4。进一步地，通过引入噪声注入（Noisy Top-K Gating）和负载均衡损失函数，MoE 有效缓解了专家利用率不均衡问题，例如 GLaM 模型以 1.2 万亿参数仅激活 97 亿参数即达到 GPT-3 的 97% 性能。当前，MoE 架构已在 GPT-4、DeepSeek、Mixtral 8x7B 等主流大模型中广泛应用，成为突破单一模型规模瓶颈的核心技术路径。

## 2.2 Prompt 工程

论文主体部分字数要求：理工类专业一般不少于 1.5 万字，其他专业一般不少于 1.0 万字。

## 2.3 本章小结

本章介绍了……

# 3 图表、公式格式

## 3.1 图表格式

本章将主要介绍一些图表和公式的格式...

表 3.1 高频感应加热的基本参数

| 感应频率（KHz） | 感应发生器功率 (%×80Kw) | 工件移动速度 (mm/min) | 感应圈与零件间隙 (mm) |
|---|---|---|---|
| 250 | 88 | 5900 | 1.65 |
| 250 | 88 | 5900 | 1.65 |
| 250 | 88 | 5900 | 1.65 |
| 250 | 88 | 5900 | 1.65 |

续表 3.1

| 感应频率（KHz） | 感应发生器功率 (%×80Kw) | 工件移动速度 (mm/min) | 感应圈与零件间隙 (mm) |
|---|---|---|---|
| 250 | 88 | 5900 | 1.65 |
| 250 | 88 | 5900 | 1.65 |

## 3.2 公式格式

$$\frac{1}{\mu}\nabla^2 A - j\omega\sigma A - \nabla(\frac{1}{\mu}) \times (\nabla \times A) + J_0 = 0 \tag{3.1}$$

## 3.3 本章小结

本章介绍了……

# 4 结论与展望

## 4.1 主要结论

本文主要……

## 4.2 研究展望

更深入的研究……

# 参考文献

[1] 杨瑞林, 李力军. 新型低合金高强韧性耐磨钢的研究 [J]. 钢铁. 1999(7): 41-45.

[2] 于潇, 刘义, 柴跃廷, 等. 互联网药品可信交易环境中主体资质审核备案模式 [J]. 清华大学学报 (自然科学版), 2012, 52(11): 1518-1523.

[3] Schinstock D.E., Cuttino J.F. Real time kinematic solutions of a non-contacting, three dimensional metrology frame[J]. Precision Engineering. 2000, 24(1): 70-76.

[4] 温诗铸. 摩擦学原理 [M]. 北京: 清华大学出版社, 1990: 296-300.

[5] 蒋有绪, 郭泉水, 马娟, 等. 中国森林群落分类及其群落学特征 [M]. 北京: 科学出版社, 1998: 5-17.

[6] 贾名字. 工程硕士论文撰写规范 [D]. 重庆: 重庆大学, 2000: 177-178.

[7] 张凯军. 轨道火车及高速轨道火车紧急安全制动辅助装置: 201220158825.2[P]. 2012-04-05.

[8] 全国信息与文献标准化技术委员会. 文献著录: 第 4 部分非书资料: GB/T 3792.4-2009[S]. 北京: 中国标准出版社, 2010: 3.

# 附录 A：XX 公式的推导

XX 公式的推导过程是：

# 致　谢

致谢主要感谢导师和对论文工作有直接贡献和帮助的人士和单位。致谢言语应谦虚诚恳，实事求是。

# 原创性声明

郑重声明：所呈交的论文（设计）《_____》，是本人在导师的指导下，独立进行研究取得的成果。除论文（设计）中已经标注引用的内容外，本论文（设计）不包含其他人或集体已经发表或撰写过的作品成果。对本文的研究做出贡献的个人和集体，均已在文中以明确方式标明。本人完全意识到本声明的法律后果，并承诺因本声明而产生的法律结果由本人承担。

论文（设计）作者签名: _____

日期：_____

# 使用授权书

本论文（设计）作者完全了解学校有关保留、使用论文（设计）的规定，同意学校保留并向国家有关部门或机构送交论文（设计）复印件和电子版，允许论文（设计）被查阅和借阅。本人授权重庆大学将本论文（设计）的全部或部分内容编入有关数据库进行检索，可以采用影印、缩印或扫描等复制方式保存和汇编本论文（设计）。

本论文（设计）属于：

保　密 □　在____ 年解密后适用本授权书

不保密 □

论文（设计）作者签名：_____　指导教师签名：_____

日期：_____　日期：_____

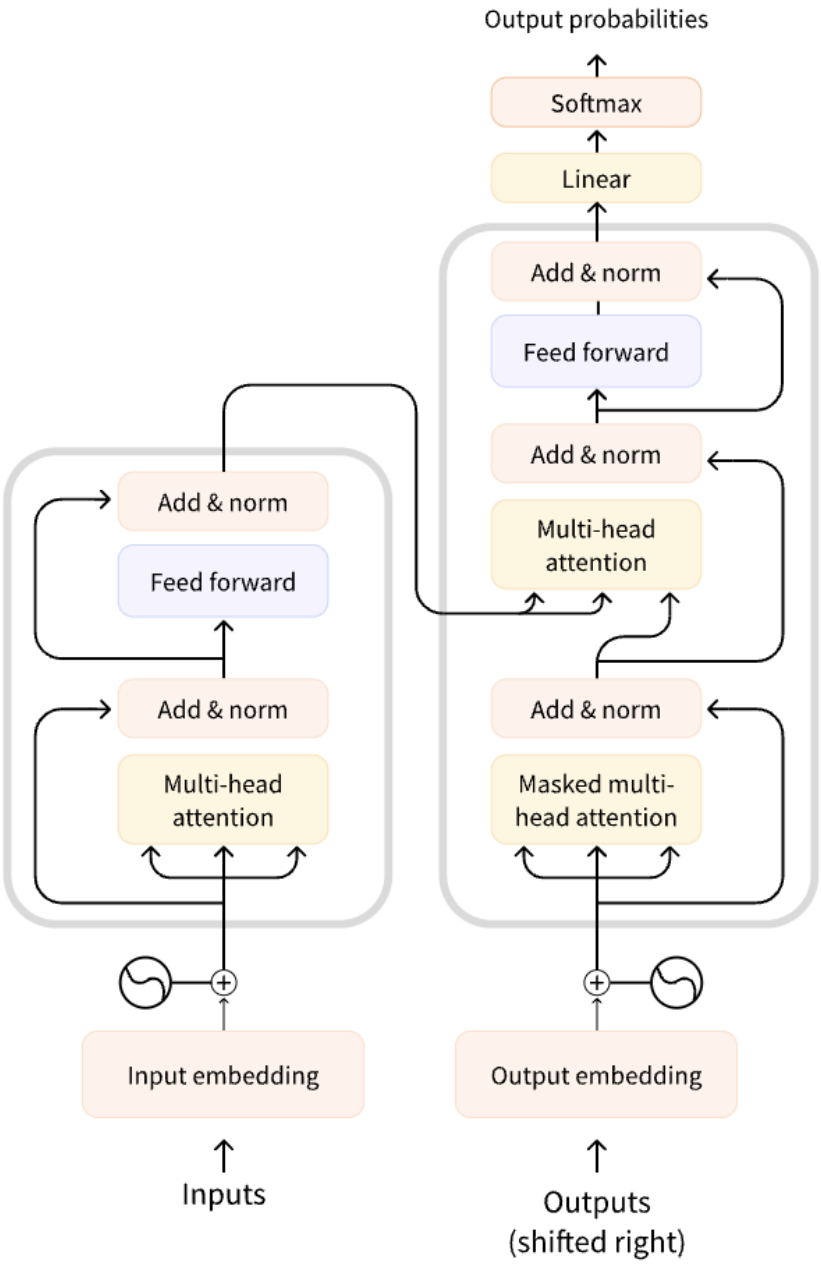


图 2.1 Transformer 结构

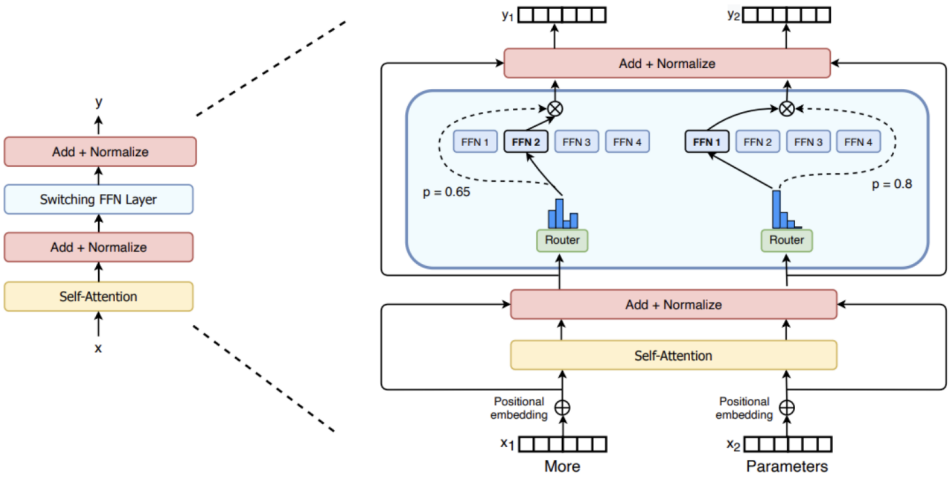


图 2.2　MoE 结构

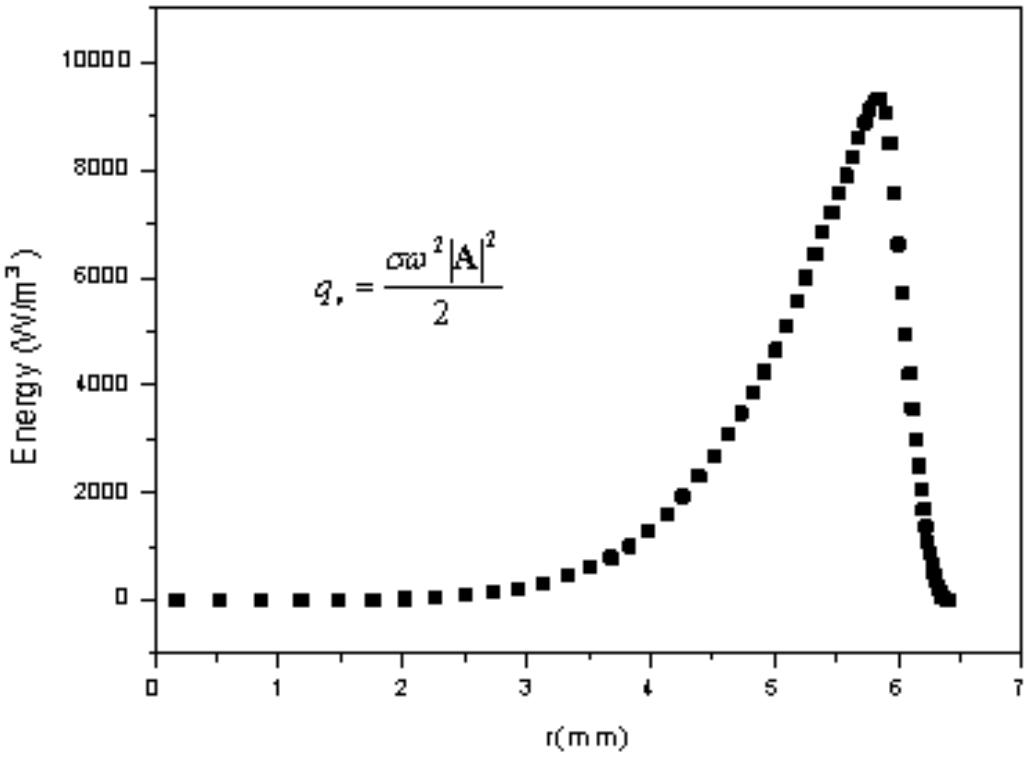


图 3.3　内热源沿径向的分布