Computer Science and Application 计算机科学与应用, 2025, 15(4), 177-185
Published Online April 2025 in Hans. https://www.hanspub.org/journal/csa
https://doi.org/10.12677/csa.2025.154090

Hans汉斯

一种基于openEuler国产操作系统的应用助手工具设计与实现

王天与，宋丽华，谭玉青，路　植

北方工业大学信息学院，北京

收稿日期：2025年3月11日；录用日期：2025年4月11日；发布日期：2025年4月18日

摘　要

国产操作系统openEuler系统中基本管理单位是rpm包，虽然rpm包为用户提供了丰富的命令和配置文件资源，但每个包中包含的命令和配置文件数量众多，且功能和用法各异，这给用户学习和使用openEuler带来了一定的困难。基于以上问题本文设计并实现了一款基于openEuler的应用助手，该应用助手构建了知识库，并实现基于TF-IDF和BM25算法的搜索功能，设计了友好的命令行界面。用户输入关键字后，助手能够智能返回相关命令和进一步的帮助信息，从而显著提升用户体验和操作效率，为openEuler社区的发展做出贡献。

关键词

openEuler，TF-IDF，BM25，操作系统，国产操作系统，应用助手

Design and Implementation of an Application Assistant Tool Based on the openEuler Domestic Operating System

Tianyu Wang, Lihua Song, Yuqing Tan, Zhi Lu

College of Information, North China University of Technology, Beijing

Received: Mar. 11th, 2025; accepted: Apr. 11th, 2025; published: Apr. 18th, 2025

Abstract

In the domestic operating system openEuler, the basic management unit is the rpm package. Although rpm packages provide users with a wealth of commands and configuration file resources, the numerous commands and configuration files contained within each package, along with their varied functionalities and usages, present a certain challenge for users in learning and using openEuler. Based on this issue, this paper designs and implements an application assistant based on openEuler. The application assistant constructs a knowledge base and implements a search functionality based

文章引用: 王天与, 宋丽华, 谭玉青, 路植. 一种基于 openEuler 国产操作系统的应用助手工具设计与实现[J]. 计算机科学与应用, 2025, 15(4): 177-185. DOI: 10.12677/csa.2025.154090

on the TF-IDF and BM25 algorithms, along with a user-friendly command-line interface. When users input keywords, the assistant can intelligently return relevant commands and further assistance information, significantly enhancing user experience and operational efficiency, thereby contributing to the development of the openEuler community.

Keywords

openEuler, TF-IDF, BM25, Operating System, Domestic Operating System, Application Assistant

1. 引言

在数字化时代，应用助手作为一种软件工具，对于提升用户工作效率、降低学习成本以及增强用户体验具有重要意义。它通过提供用户界面、指导和建议等方式，简化了用户与操作系统或应用程序之间的交互，成为用户与系统之间的桥梁。应用助手不仅能够提高学习效率、降低成本，还能推动技术推广与创新，拓展行业应用，挖掘数据价值，并提供安全便捷的服务。随着人们对信息获取和问题解决效率的不断追求，具备智能问答系统的应用助手逐渐成为了满足用户需求的重要工具[1]。

然而，在 openEuler [2]操作系统中，用户面临着诸多问题和挑战。openEuler 是一个开源的 Linux 发行版，支持服务器、云计算、边缘计算、嵌入式等多种应用场景，其基本管理单位是 rpm 包[3]。虽然 rpm 包为用户提供了丰富的命令和配置文件资源，但每个包中包含的命令和配置文件数量众多，且功能和用法各异，这给用户学习和使用 openEuler 带来了一定的困难。用户在寻找特定功能对应的命令时可能会遇到障碍，需要一个能够将所需功能与具体命令相连接的工具，以帮助他们更高效地使用 openEuler。

本研究旨在解决 openEuler 操作系统用户在学习和使用过程中面临的复杂性和效率问题。由于 openEuler 作为基于 rpm 包管理的 Linux 发行版，包含大量的配置文件、包和命令，这对初学者造成困难，尤其是在寻找特定功能对应的命令时，用户容易感到困惑。此外，缺乏有效的工具支持以及现有知识库更新滞后，导致用户在使用新功能时的信息不足。因此，本研究将使用 AI 技术开发一个智能搜索助手工具，通过计算用户需求描述信息与 openEuler 系统提供的相关命令信息，建立知识库，用户交互获取 AI 计算后的结果信息，从而降低学习难度，提高操作效率，促进 openEuler 的广泛应用和用户满意度。

本项目的目标是开发一款智能助手工具，用于提升用户在使用 openEuler 操作系统时的效率和体验。具体说来，本工具将作为一个主机范围内的搜索引擎，当用户输入几个关键字后，能够智能地返回与这些关键字可能相关的命令以及进一步的帮助信息。通过这种方式，用户可以快速找到实现特定功能所需的命令和配置文件，从而降低学习难度，提高操作效率。这款工具的集成和升级设计也将确保其能够随时更新，保持信息的时效性，为用户提供持续的支持。

为了实现这一目标，本研究将遵循以下技术路线：首先，构建知识库，系统地收集 openEuler 系统中所有已知的 rpm 包，从每个 rpm 包中提取出包含的命令和配置文件，并将收集到的信息整理成一个结构化的知识库。其次，开发关键字搜索功能，实现基于关键字的搜索算法，根据匹配度对搜索结果进行排序，并为每个匹配的命令和配置文件生成简要描述。而后，进行工具集成与分离设计，确保智能助手工具能够高效地访问和使用知识库，同时允许知识库在不影响工具运行的情况下进行升级和维护。此外，设计简洁直观的用户界面和优化用户与工具的交互流程，使用户能够轻松地输入关键字并查看搜索结果。

最后，进行测试与优化，对智能助手工具的各个功能进行测试，根据测试结果进行性能优化，并收集用户反馈，根据用户的实际使用体验对工具进行调整和优化。

通过本研究，希望能够为 openEuler 操作系统用户提供一个更加友好、高效、智能的使用环境，促进 openEuler 的广泛应用，推动整个开源社区的发展，并为未来数字基础设施的演变做出积极贡献。

2. 系统设计

2.1. 知识库构建

在知识库构建过程中，首先需系统地收集 openEuler 系统中所有已知的 rpm 包信息。随后，从每个 rpm 包中提取出包含的命令和配置文件，最后将收集到的命令和配置文件信息整理成一个结构化的知识库，以便后续的搜索和查询。

2.2. 关键字搜索功能开发

开发基于关键字的搜索算法，能够快速从知识库中检索出与用户输入关键字相关的命令和配置文件。同时，实现一个匹配度排序机制，确保最相关的结果排在前面，并为每个匹配的命令和配置文件生成简要描述，帮助用户快速理解其功能和用途。

2.3. 工具集成与分离设计

在设计智能助手工具与知识库的集成方式，使其能够高效地访问和使用知识库，同时设计知识库的独立更新机制，允许在不影响工具运行的情况下进行升级和维护。

2.4. 用户界面和交互设计

设计一个简洁直观的用户界面，方便用户输入关键字并查看搜索结果。优化用户与工具的交互流程，确保用户能够快速获得所需信息，并提供进一步的操作指引。

2.5. 测试与优化

对智能助手工具的各个功能进行测试，确保其能够准确返回相关命令和帮助信息。根据测试结果对工具进行性能优化，提高搜索速度和准确性，并收集用户反馈，根据用户的实际使用体验对工具进行调整和改进。

3. 系统实现

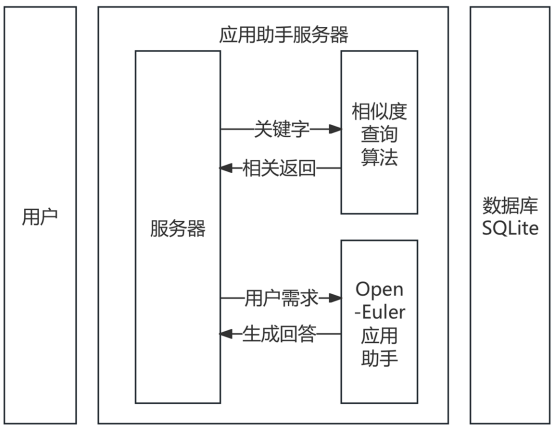

Figure 1. System framework diagram
图 1. 系统框架图

图 1 展示了应用助手服务器与用户、数据库 SQLite 之间的交互流程。

3.1. 环境与配置实现

● 操作系统与 Python 版本：在 openEuler 22.03 (LTS-SP3)和 Windows 11 双系统环境下，安装 Python 3.9.9，确保开发与运行环境的一致性和稳定性，为后续的开发工作提供坚实基础。

● 依赖库安装：通过 Python 的包管理工具 pip，依次安装 click 8.1.7、Colorama 0.4.6、nltk 3.8.1 等依赖库，各库版本严格对应，避免版本冲突导致的功能异常，保障系统各功能模块的正常调用与协同工作。

● 代码编辑与数据配置：利用 VSCode 远程连接进行代码编辑，借助其强大的代码提示、调试等功能，提高开发效率。数据源方面，以 openEuler 虚拟机中的 rpm 包信息为基础，从 http://man.he.net/爬取 rpm 包所包含的指令信息，数据格式采用 db 和 csv，选用 SQLite 数据库进行存储，实现数据的高效管理和便捷访问。

3.2. 模型实现

3.2.1. 模型概述与组件

通过模型计算文本相似度检索与用户输入相似的软件包指令，采用 TF-IDF 向量化和余弦相似度度量方法。其中，TF-IDF 向量化器负责将文本转换为特征向量，余弦相似度用于计算向量间的相似度，相关文件如 tfidf_vectorizer.pickle 存储 TF-IDF 模型，tfidf_matrix.pickle 存储软件包指令的 TF-IDF 向量。

3.2.2. 训练策略

● 数据预处理：借助 Pandas 库读取 inst_data.csv 文件，对数据进行清洗、格式化等预处理操作，确保数据质量。

● 向量化：利用 pickle 加载的 TF-IDF 向量化器，将用户输入文本转换为 TF-IDF 特征向量，实现文本的数值化表示。

● 相似度计算：调用 cosine_similarity 函数，计算用户输入向量与预存的 TF-IDF 矩阵中所有向量的余弦相似度，量化文本间的相似性。

● 结果排序：依据相似度分数对搜索结果进行降序排序，将最相关的软件包指令优先展示给用户。

3.2.3. 搜索策略

● 相似度检索：search 函数调用 tfidf 函数，获取与用户输入相似度最高的软件包指令 ID 及对应分数，快速定位目标指令。

● 信息检索：借助 access_inst.SelectBriefInfoByIds 函数，根据指令 ID 从数据库中检索软件包的简要信息，丰富搜索结果的内容。

● 综合排序：结合软件包的版本号和相似度分数，对结果进行综合排序，既考虑指令的相关性，又兼顾软件的新旧程度，为用户提供更合理的搜索结果。

3.3. 性能优化实现

虽未明确提及自动混合精度，但借鉴深度学习模型中的优化思路，可考虑在后续优化中引入，以减少 GPU 显存使用并提高训练速度。同时，对现有代码进行分析，查找潜在的性能瓶颈，如循环结构、数据库查询等，通过算法优化、索引建立等方式，提升系统的整体运行效率。

3.4. 数据实现

3.4.1. 数据集描述与收集

安装 openEuler 虚拟机，利用命令行工具获取本地 rpm 包信息，包括包名、版本号、文件路径等，同

时从 http://man.he.net/爬取详细指令信息，将收集到的数据保存至数据库，构建完整、准确的数据集，为模型训练和系统运行提供数据支撑。

3.4.2. 数据处理

在数据处理方面，首先，通过执行 rpm -qa 命令获取已安装的 rpm 包列表，随后解码并分割结果，以移除空字符串，从而得到有效的包名列表。针对每个 rpm 包，接着使用 rpm -ql instname 命令获取其包含的文件路径，筛选出以/bin/和/sbin/开头的路径，从中提取用户指令和管理员指令名称，这为后续的指令信息整合与分析奠定了基础。

3.4.3. 数据库设计

在数据库设计中，构建了一个关系模型，如图 2 数据 ER 图所示。rpm 表以 id 作为主键，存储包名、版本号、架构等详细信息，全面描述每个 rpm 包的属性。同时，inst 表也以 id 为主键，记录安装实例的名称、描述、简介等元数据，并关联相应的 rpm 包和评分信息，便于用户快速了解指令的基本情况。此外，opt 表通过 id 唯一标识每个选项，并关联 inst 表的 inst_id，存储选项名称和内容，以提供丰富的指令配置信息。这种设计将数据结构化，便于后续查询与操作。

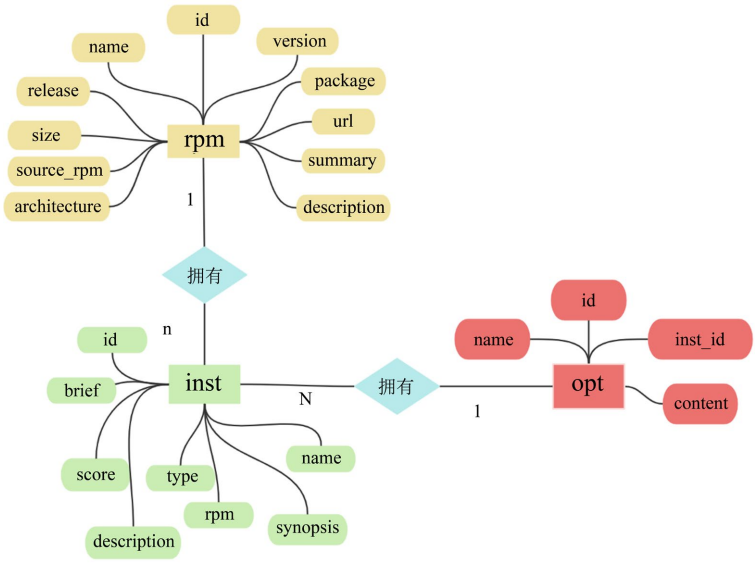

Figure 2. Entity-relationship diagram
图 2. 数据 ER 图

- access_rpm：负责与 rpm 表交互，实现 rpm 包信息的查询、插入、更新等操作，为系统提供底层的数据支持。
- access_inst：专注于 inst 表的数据访问，支持安装实例的检索、筛选等功能，满足用户对指令信息的多样化查询需求。
- access_opt：管理 opt 表的数据操作，方便用户获取指令的详细配置选项，增强系统的实用性和易用性。

3.5. 搜索排序算法实现

3.5.1. TF-IDF [4]与余弦相似度计算

1) TF-IDF 原理与公式

- 词频(TF)：计算词在文档中出现的次数除以文档中总词数，反映词在文档中的重要性。

$$TF\left(t,d\right)=\frac{\text{词}\mathbf{t}\text{在文档}\mathbf{d}\text{中的出现次数}}{\text{文档}\mathbf{d}\text{中的总词数}} \tag{1}$$

- 逆文档频率(IDF)：衡量词在语料库中的普遍重要性，通过语料库文档总数除以包含该词文档数再取对数得到。

$$IDF\left(t,D\right)=\log\left(\frac{\text{文档集合}\boldsymbol{D}\text{的总文档数}}{\text{包含词}\boldsymbol{t}\text{的文档数}}\right) \tag{2}$$

- TF-IDF 值：词在文档中的重要性通过 TF 和 IDF 的乘积衡量。

$$TF-IDF\left(t,d,D\right)=TF\left(t,d\right)\times IDF\left(t,D\right) \tag{3}$$

2) 余弦相似度计算

余弦相似度是通过计算两个向量的点积和它们模的乘积的比值来确定的。对于两个向量 A 和 B，余弦相似度的计算公式为：

$$cosine\,similarity=\frac{A\cdot B}{\left\|\mathrm{A}\right\|\times\left\|\mathrm{B}\right\|} \tag{4}$$

其中，A·B 是向量 A 和 B 的点积，$\left\|\mathrm{A}\right\|$ 和 $\left\|\mathrm{B}\right\|$ 分别是向量 A 和 B 的模。

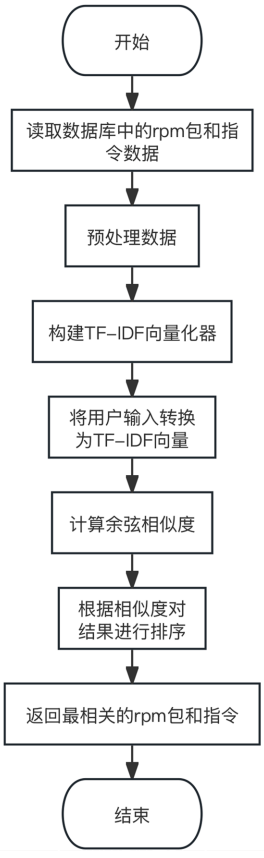

Figure 3. System search flowchart
图 3. 系统搜索流程图

如图 3 所示为基于 TF-IDF 的应用助手工具的工作流程图。

3.5.2. BM25 [5]算法实现

1) BM25 原理与公式

- IDF 计算：IDF 也就是逆文档频率，计算公式如下：

$$IDF\left(q_i\right)=\log\left(\frac{N-n\left(q_i\right)+0.5}{n\left(q_i\right)+0.5}\right) \tag{5}$$

N 为文档总数，$n\ (q_i)$为包含词 q_i 的文档数，通过调整公式，使常见词的权重降低，突出稀有词的重要性。

- R 相关性得分：$R\ (q_i, d)$相关性得分的一般性公式：

$$R\left(q_i,d\right)=\frac{f_i\cdot\left(k_1+1\right)}{f_i+K}\cdot\frac{q_{f_i}\cdot\left(k_2+1\right)}{q_{f_i}+k_2} \tag{6}$$

$$K=k_1\cdot\left(1-b+b\cdot\frac{dl}{avgdl}\right) \tag{7}$$

f_i 为 q_i 在文档 d 中的频率，q_{fi} 为 q_i 在输入句子中的频率，dl 为文档长度，$avgdl$ 为平均长度，通过调节因子 k_1、k_2、b，实现对词频和文档长度的灵活调整，更精准地评估文档与查询的相关性。

- BM25 得分汇总：

$$Score\left(D,Q\right)=\sum_{i=1}^{n}IDF\left(q_i\right)\cdot\frac{f\left(q_i,D\right)\cdot\left(k_1+1\right)}{f\left(q_i,D\right)+k_1\cdot\left(1-b+b\cdot\frac{avgdl}{|D|}\right)} \tag{8}$$

综合考虑各词的相关性得分，得出文档与查询的整体匹配程度。

2) BM25 代码实现

读取 CSV 数据，加载文本数据并转化为词项列表，初始化 BM25Okapi 对象，将用户输入转化为词项列表，计算用户输入与所有文档的 BM25 得分，依据得分排序获取相关度高的文档 ID 及对应分数，实现精准的文档检索与排序。

3.6. UI 界面设计实现

设计思路与组件使用遵循简洁性、可用性、灵活性的设计思路，采用基于命令行的交互式界面，利用 npyscreen 库构建，整合 SearchBox、SearchResults、InstructSelect、Detail 等组件，分别实现搜索关键字输入、搜索结果展示、指令选项呈现、详细信息查看等功能，通过键盘操作实现界面间的顺畅切换与信息浏览。

具体实现交互过程是用户通过 SearchBox 中输入搜索关键词。按下回车键后，系统将触发搜索，并展示相关结果。

搜索结果在 SearchResults 组件中显示，用户可以使用方向键上下移动，选择不同的搜索结果。同时采用 MultiLineAction 组件使选择过程更加直观，通过简单的键盘操作即可快速定位所需信息。

另外还提供了快捷键操作：

- q：快速退出程序，便于在完成操作后迅速结束会话。
- w：返回到上一个界面，减少误操作造成的困扰。
- j 和 k：分别用于在搜索结果和选项中上下滚动，提供更灵活的浏览体验。

4. 实验分析

上述部分提到的 TF-IDF 和 BM25 是信息检索中常用的两种算法，它们在处理词频时有所不同。TF-

IDF 的词频饱和度是线性的，即词频的增加会直接导致得分的线性增长。然而，这种线性增长方式可能导致高频词对得分的过度影响，尤其是在长文档中，这种问题更为突出。

相比之下 BM25 采用非线性饱和度，随着词频增加，得分增长逐渐放缓，有助于避免高频词对得分的过度影响，尤其在长文档中更为稳定。BM25 通过参数 k 和 b 来控制词频饱和度和文档长度归一化，使得评估更准确。参数 k(默认 1.2)控制饱和度变化速度，参数 b(默认 0.75)控制归一化程度。这种灵活的参数调节机制使得 BM25 在处理长文档时表现更为出色，能够提供更稳定的相关性评分。

图 4 展示了两种模型的词频饱和度对比。TF-IDF 得分随词频线性增长，而 BM25 增长放缓，最终平稳。这表明 BM25 在处理词频时更为稳健，适合高质量检索场景。

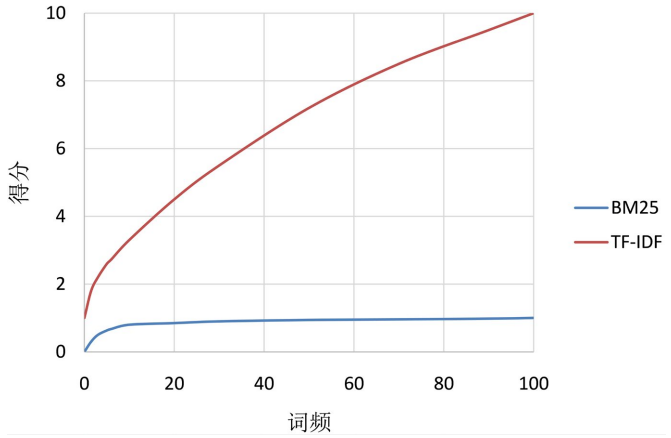

Figure 4. Term frequency saturation of TF-IDF and BM25 models [6]
图 4. TF-IDF 模型与 BM25 模型的词频饱和度[6]

对 TF-IDF 模型与 BM25 模型在信息检索任务中的表现进行了对比分析，涉及词频处理、文档长度影响、可扩展性、计算复杂度、效果、适用场景和鲁棒性等七个方面。

表 1 展示了两种模型在这些特性上的表现差异，有助于深入理解每种模型的优势及其适用场景。

Table 1. Comparison of TF-IDF Model and BM25 Model
表 1. TF-IDF 模型与 BM25 模型对比

特性/算法	TF-IDF	BM25
词频的处理	词频越高，重要性越大，可能导致长文档被偏高评分	引入非线性词频变换，避免词频过高的词占主导地位
文档长度影响	不考虑文档的长度，可能导致长文档获得过高的权重	引入了文档长度正则化项，减少了文档长度对结果的影响，限制了 TF 值的极限增长
可扩展性	可扩展性一般，对于大规模数据集可能需要额外的优化	可扩展性好，易于与其他算法和技术结合使用
计算复杂度	计算较为简单适用于快速计算	考虑了词频的平滑和文档长度的正则化，计算稍微复杂
效果	对文档内容的直观度量，效果有限	对文档长度和词频进行平衡，能提供更准确的排名
适用场景	尤其适合小规模文档集信息检索	更加复杂和精确，适用于需要高质量检索结果的场景
鲁棒性	对于不同长度的文档，鲁棒性较差	对于不同长度的文档，鲁棒性较好，能够提供更稳定的相关性评分

5. 结论

本文针对国产操作系统 openEuler 用户在学习和使用过程中面临的挑战，提出了一款基于 openEuler 的应用助手。通过构建知识库并实现基于 TF-IDF 和 BM25 算法的搜索功能，该助手能够有效帮助用户快速搜索到所需的命令和配置文件。实验分析表明，BM25 算法在处理词频和文档长度方面表现更优，能够提供更准确的搜索结果和更稳定的相关性评分，尤其适用于需要高质量检索结果的场景。

本研究为 openEuler 用户提供了一个高效、智能的使用环境，推动了 openEuler 社区的发展。未来的工作将集中在进一步优化搜索算法，扩展知识库的覆盖范围，并探索更多用户交互方式，以满足不断增长的用户需求。此外，我们也将关注智能助手在其他国产操作系统中的应用潜力，期待为更广泛的用户提供支持。

基金项目

大学生创新创业训练计划课题(2025)；北方工业大学教育教学改革项目。

参考文献

[1] 陈冬雷, 秦薇. 基于检索增强生成的智能问答系统关键技术研究[J]. 办公自动化, 2024, 29(19): 82-86.

[2] Zhou, M., Hu, X. and Xiong, W. (2022) Openeuler: Advancing a Hardware and Software Application Ecosystem. *IEEE Software*, **39**, 101-105. https://doi.org/10.1109/ms.2021.3132138

[3] 陆志烽. Linux 下 Samba 服务器的搭建——实现 LINUX 与 WINDOWS 之间文件共享[J]. 科技信息, 2012(9): 86.

[4] Ramos, J. (2003) Using TF-IDF to Determine Word Relevance in Document Queries. *Proceedings of the First Instructional Conference on Machine Learning*, **242**, 29-48.

[5] Robertson, S. and Zaragoza, H. (2009) The Probabilistic Relevance Framework: BM25 and Beyond. *Foundations and Trends® in Information Retrieval*, **3**, 333-389. https://doi.org/10.1561/1500000019

[6] seetimee. TF-IDF 和 BM25 原理和区别[EB/OL]. https://blog.csdn.net/stephen147/article/details/140117738, 2025-02-12.