

本文的公有所有声明

Public-owned Open-source General Public Statement

By author

本文作者基于公有所有开源项目编写，并遵循开源协议 GNU GPL 3.0 这种公有所有协议。

This software is based on public-owned open-source local project, so does the document obeys GNU GPL public-owned 3.0 license.

Author: Fu Chensheng.

GNU GENERAL PUBLIC LICENSE

Version 3, 29 June 2007

Copyright © 2007 Free Software Foundation, Inc. <<https://fsf.org/>>

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

Preamble

The GNU General Public License is a free, copyleft license for software and other kinds of works.

The licenses for most software and other practical works are designed to take away your freedom to share and change the works. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change all versions of a program--to make sure it remains free software for all its users. We, the Free Software Foundation, use the GNU General Public License for most of our software; it applies also to any other work released this way by its authors. You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for them if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs, and that you know you can do these things.

To protect your rights, we need to prevent others from denying you these rights or asking you to surrender the rights. Therefore, you have certain responsibilities if you distribute copies of the software, or if you modify it: responsibilities to respect the freedom of others.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must pass on to the recipients the same freedoms that you received. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

Developers that use the GNU GPL protect your rights with two steps: (1) assert copyright on the software, and (2) offer you this License giving you legal permission to copy, distribute and/or modify it.

For the developers' and authors' protection, the GPL clearly explains that there is no warranty for this free software. For both users' and authors' sake, the GPL requires that modified versions be marked as changed, so that their problems will not be attributed erroneously to authors of previous versions.

Some devices are designed to deny users access to install or run modified versions of the software inside them, although the manufacturer can do so. This is fundamentally incompatible with the aim of protecting users' freedom to change the software. The systematic pattern of such abuse occurs in the area of products for individuals to use, which is precisely where it is most unacceptable. Therefore, we have designed this version of the GPL to prohibit the practice for those products. If such problems arise substantially in other domains, we stand ready to extend this provision to those domains in future versions of the GPL, as needed to protect the freedom of users.

Finally, every program is threatened constantly by software patents. States should not allow patents to restrict development and use of software on general-purpose computers, but in those that do, we wish to avoid the special danger that patents applied to a free program could make it effectively proprietary. To prevent this, the GPL assures that patents cannot be used to render the program non-free.

The precise terms and conditions for copying, distribution and modification follow.

TERMS AND CONDITIONS

0. Definitions.

“This License” refers to version 3 of the GNU General Public License.

“Copyright” also means copyright-like laws that apply to other kinds of works, such as semiconductor masks.

“The Program” refers to any copyrightable work licensed under this License. Each licensee is addressed as “you”. “Licensees” and “recipients” may be individuals or organizations.

To “modify” a work means to copy from or adapt all or part of the work in a fashion requiring copyright permission, other than the making of an exact copy. The resulting work is called a “modified version” of the earlier work or a work “based on” the earlier work.

A “covered work” means either the unmodified Program or a work based on the Program.

To “propagate” a work means to do anything with it that, without permission, would make you directly or secondarily liable for infringement under applicable copyright law, except executing it on a computer or modifying a private copy. Propagation includes copying, distribution (with or without modification), making available to the public, and in some countries other activities as well.

To “convey” a work means any kind of propagation that enables other parties to make or receive copies. Mere interaction with a user through a computer network, with no transfer of a copy, is not conveying.

An interactive user interface displays “Appropriate Legal Notices” to the extent that it includes a convenient and prominently visible feature that (1) displays an appropriate copyright notice, and (2) tells the user that there is no warranty for the work (except to the extent that warranties are provided), that licensees may convey the work under this License, and how to view a copy of this License. If the interface presents a list of user commands or options, such as a menu, a prominent item in the list meets this criterion.

1. Source Code.

The “source code” for a work means the preferred form of the work for making modifications to it. “Object code” means any non-source form of a work.

A “Standard Interface” means an interface that either is an official standard defined by a recognized standards body, or, in the case of interfaces specified for a particular programming language, one that is widely used among developers working in that language.

The “System Libraries” of an executable work include anything, other than the work as a whole, that (a) is included in the normal form of packaging a Major Component, but which is not part of that Major Component, and (b) serves only to enable use of the work with that Major Component, or to implement a Standard Interface for which an implementation is available to the public in source code form. A “Major Component”, in this context, means a major essential component (kernel, window system, and so on) of the specific operating system (if any) on which the executable work runs, or a compiler used to produce the work, or an object code interpreter used to run it.

The “Corresponding Source” for a work in object code form means all the source code needed to generate, install, and (for an executable work) run the object code and to modify the work, including scripts to control those activities. However, it does not include the work's System Libraries, or general-purpose tools or generally available free programs which are used unmodified in performing those activities but which are not part of the work. For example, Corresponding Source includes interface definition files associated with source files for the work, and the source code for shared libraries and dynamically linked subprograms that the work is specifically designed to require, such as by intimate data communication or control flow between those subprograms and other parts of the work.

The Corresponding Source need not include anything that users can regenerate automatically from other parts of the Corresponding Source.

The Corresponding Source for a work in source code form is that same work.

2. Basic Permissions.

All rights granted under this License are granted for the term of copyright on the Program, and are irrevocable provided the stated conditions are met. This License

explicitly affirms your unlimited permission to run the unmodified Program. The output from running a covered work is covered by this License only if the output, given its content, constitutes a covered work. This License acknowledges your rights of fair use or other equivalent, as provided by copyright law.

You may make, run and propagate covered works that you do not convey, without conditions so long as your license otherwise remains in force. You may convey covered works to others for the sole purpose of having them make modifications exclusively for you, or provide you with facilities for running those works, provided that you comply with the terms of this License in conveying all material for which you do not control copyright. Those thus making or running the covered works for you must do so exclusively on your behalf, under your direction and control, on terms that prohibit them from making any copies of your copyrighted material outside their relationship with you.

Conveying under any other circumstances is permitted solely under the conditions stated below. Sublicensing is not allowed; section 10 makes it unnecessary.

3. Protecting Users' Legal Rights From Anti-Circumvention Law.

No covered work shall be deemed part of an effective technological measure under any applicable law fulfilling obligations under article 11 of the WIPO copyright treaty adopted on 20 December 1996, or similar laws prohibiting or restricting circumvention of such measures.

When you convey a covered work, you waive any legal power to forbid circumvention of technological measures to the extent such circumvention is effected by exercising rights under this License with respect to the covered work, and you disclaim any intention to limit operation or modification of the work as a means of enforcing, against the work's users, your or third parties' legal rights to forbid circumvention of technological measures.

4. Conveying Verbatim Copies.

You may convey verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice; keep intact all notices stating that this License and any non-permissive terms added in accord with section 7 apply to the

code; keep intact all notices of the absence of any warranty; and give all recipients a copy of this License along with the Program.

You may charge any price or no price for each copy that you convey, and you may offer support or warranty protection for a fee.

5. Conveying Modified Source Versions.

You may convey a work based on the Program, or the modifications to produce it from the Program, in the form of source code under the terms of section 4, provided that you also meet all of these conditions:

- a) The work must carry prominent notices stating that you modified it, and giving a relevant date.
- b) The work must carry prominent notices stating that it is released under this License and any conditions added under section 7. This requirement modifies the requirement in section 4 to “keep intact all notices”.
- c) You must license the entire work, as a whole, under this License to anyone who comes into possession of a copy. This License will therefore apply, along with any applicable section 7 additional terms, to the whole of the work, and all its parts, regardless of how they are packaged. This License gives no permission to license the work in any other way, but it does not invalidate such permission if you have separately received it.
- d) If the work has interactive user interfaces, each must display Appropriate Legal Notices; however, if the Program has interactive interfaces that do not display Appropriate Legal Notices, your work need not make them do so.

A compilation of a covered work with other separate and independent works, which are not by their nature extensions of the covered work, and which are not combined with it such as to form a larger program, in or on a volume of a storage or distribution medium, is called an “aggregate” if the compilation and its resulting copyright are not used to limit the access or legal rights of the compilation’s users beyond what the individual works permit. Inclusion of a covered work in an aggregate does not cause this License to apply to the other parts of the aggregate.

6. Conveying Non-Source Forms.

You may convey a covered work in object code form under the terms of sections 4 and 5, provided that you also convey the machine-readable Corresponding Source under the terms of this License, in one of these ways:

- a) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by the Corresponding Source fixed on a durable physical medium customarily used for software interchange.
- b) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by a written offer, valid for at least three years and valid for as long as you offer spare parts or customer support for that product model, to give anyone who possesses the object code either (1) a copy of the Corresponding Source for all the software in the product that is covered by this License, on a durable physical medium customarily used for software interchange, for a price no more than your reasonable cost of physically performing this conveying of source, or (2) access to copy the Corresponding Source from a network server at no charge.
- c) Convey individual copies of the object code with a copy of the written offer to provide the Corresponding Source. This alternative is allowed only occasionally and noncommercially, and only if you received the object code with such an offer, in accord with subsection 6b.
- d) Convey the object code by offering access from a designated place (gratis or for a charge), and offer equivalent access to the Corresponding Source in the same way through the same place at no further charge. You need not require recipients to copy the Corresponding Source along with the object code. If the place to copy the object code is a network server, the Corresponding Source may be on a different server (operated by you or a third party) that supports equivalent copying facilities, provided you maintain clear directions next to the object code saying where to find the Corresponding Source. Regardless of what server hosts the Corresponding Source, you remain obligated to ensure that it is available for as long as needed to satisfy these requirements.
- e) Convey the object code using peer-to-peer transmission, provided you inform other peers where the object code and Corresponding Source of the

work are being offered to the general public at no charge under subsection 6d.

A separable portion of the object code, whose source code is excluded from the Corresponding Source as a System Library, need not be included in conveying the object code work.

A “User Product” is either (1) a “consumer product”, which means any tangible personal property which is normally used for personal, family, or household purposes, or (2) anything designed or sold for incorporation into a dwelling. In determining whether a product is a consumer product, doubtful cases shall be resolved in favor of coverage. For a particular product received by a particular user, “normally used” refers to a typical or common use of that class of product, regardless of the status of the particular user or of the way in which the particular user actually uses, or expects or is expected to use, the product. A product is a consumer product regardless of whether the product has substantial commercial, industrial or non-consumer uses, unless such uses represent the only significant mode of use of the product.

“Installation Information” for a User Product means any methods, procedures, authorization keys, or other information required to install and execute modified versions of a covered work in that User Product from a modified version of its Corresponding Source. The information must suffice to ensure that the continued functioning of the modified object code is in no case prevented or interfered with solely because modification has been made.

If you convey an object code work under this section in, or with, or specifically for use in, a User Product, and the conveying occurs as part of a transaction in which the right of possession and use of the User Product is transferred to the recipient in perpetuity or for a fixed term (regardless of how the transaction is characterized), the Corresponding Source conveyed under this section must be accompanied by the Installation Information. But this requirement does not apply if neither you nor any third party retains the ability to install modified object code on the User Product (for example, the work has been installed in ROM).

The requirement to provide Installation Information does not include a requirement to continue to provide support service, warranty, or updates for a work that has been modified or installed by the recipient, or for the User Product in which it has been modified or installed. Access to a network may be denied when the

modification itself materially and adversely affects the operation of the network or violates the rules and protocols for communication across the network.

Corresponding Source conveyed, and Installation Information provided, in accord with this section must be in a format that is publicly documented (and with an implementation available to the public in source code form), and must require no special password or key for unpacking, reading or copying.

7. Additional Terms.

“Additional permissions” are terms that supplement the terms of this License by making exceptions from one or more of its conditions. Additional permissions that are applicable to the entire Program shall be treated as though they were included in this License, to the extent that they are valid under applicable law. If additional permissions apply only to part of the Program, that part may be used separately under those permissions, but the entire Program remains governed by this License without regard to the additional permissions.

When you convey a copy of a covered work, you may at your option remove any additional permissions from that copy, or from any part of it. (Additional permissions may be written to require their own removal in certain cases when you modify the work.) You may place additional permissions on material, added by you to a covered work, for which you have or can give appropriate copyright permission.

Notwithstanding any other provision of this License, for material you add to a covered work, you may (if authorized by the copyright holders of that material) supplement the terms of this License with terms:

- a) Disclaiming warranty or limiting liability differently from the terms of sections 15 and 16 of this License; or
- b) Requiring preservation of specified reasonable legal notices or author attributions in that material or in the Appropriate Legal Notices displayed by works containing it; or
- c) Prohibiting misrepresentation of the origin of that material, or requiring that modified versions of such material be marked in reasonable ways as different from the original version; or
- d) Limiting the use for publicity purposes of names of licensors or authors of the material; or

- e) Declining to grant rights under trademark law for use of some trade names, trademarks, or service marks; or
- f) Requiring indemnification of licensors and authors of that material by anyone who conveys the material (or modified versions of it) with contractual assumptions of liability to the recipient, for any liability that these contractual assumptions directly impose on those licensors and authors.

All other non-permissive additional terms are considered “further restrictions” within the meaning of section 10. If the Program as you received it, or any part of it, contains a notice stating that it is governed by this License along with a term that is a further restriction, you may remove that term. If a license document contains a further restriction but permits relicensing or conveying under this License, you may add to a covered work material governed by the terms of that license document, provided that the further restriction does not survive such relicensing or conveying.

If you add terms to a covered work in accord with this section, you must place, in the relevant source files, a statement of the additional terms that apply to those files, or a notice indicating where to find the applicable terms.

Additional terms, permissive or non-permissive, may be stated in the form of a separately written license, or stated as exceptions; the above requirements apply either way.

8. Termination.

You may not propagate or modify a covered work except as expressly provided under this License. Any attempt otherwise to propagate or modify it is void, and will automatically terminate your rights under this License (including any patent licenses granted under the third paragraph of section 11).

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this

License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, you do not qualify to receive new licenses for the same material under section 10.

9. Acceptance Not Required for Having Copies.

You are not required to accept this License in order to receive or run a copy of the Program. Ancillary propagation of a covered work occurring solely as a consequence of using peer-to-peer transmission to receive a copy likewise does not require acceptance. However, nothing other than this License grants you permission to propagate or modify any covered work. These actions infringe copyright if you do not accept this License. Therefore, by modifying or propagating a covered work, you indicate your acceptance of this License to do so.

10. Automatic Licensing of Downstream Recipients.

Each time you convey a covered work, the recipient automatically receives a license from the original licensors, to run, modify and propagate that work, subject to this License. You are not responsible for enforcing compliance by third parties with this License.

An “entity transaction” is a transaction transferring control of an organization, or substantially all assets of one, or subdividing an organization, or merging organizations. If propagation of a covered work results from an entity transaction, each party to that transaction who receives a copy of the work also receives whatever licenses to the work the party's predecessor in interest had or could give under the previous paragraph, plus a right to possession of the Corresponding Source of the work from the predecessor in interest, if the predecessor has it or can get it with reasonable efforts.

You may not impose any further restrictions on the exercise of the rights granted or affirmed under this License. For example, you may not impose a license fee, royalty, or other charge for exercise of rights granted under this License, and you may not initiate litigation (including a cross-claim or counterclaim in a lawsuit)

alleging that any patent claim is infringed by making, using, selling, offering for sale, or importing the Program or any portion of it.

11. Patents.

A “contributor” is a copyright holder who authorizes use under this License of the Program or a work on which the Program is based. The work thus licensed is called the contributor's “contributor version”.

A contributor's “essential patent claims” are all patent claims owned or controlled by the contributor, whether already acquired or hereafter acquired, that would be infringed by some manner, permitted by this License, of making, using, or selling its contributor version, but do not include claims that would be infringed only as a consequence of further modification of the contributor version. For purposes of this definition, “control” includes the right to grant patent sublicenses in a manner consistent with the requirements of this License.

Each contributor grants you a non-exclusive, worldwide, royalty-free patent license under the contributor's essential patent claims, to make, use, sell, offer for sale, import and otherwise run, modify and propagate the contents of its contributor version.

In the following three paragraphs, a “patent license” is any express agreement or commitment, however denominated, not to enforce a patent (such as an express permission to practice a patent or covenant not to sue for patent infringement). To “grant” such a patent license to a party means to make such an agreement or commitment not to enforce a patent against the party.

If you convey a covered work, knowingly relying on a patent license, and the Corresponding Source of the work is not available for anyone to copy, free of charge and under the terms of this License, through a publicly available network server or other readily accessible means, then you must either (1) cause the Corresponding Source to be so available, or (2) arrange to deprive yourself of the benefit of the patent license for this particular work, or (3) arrange, in a manner consistent with the requirements of this License, to extend the patent license to downstream recipients. “Knowingly relying” means you have actual knowledge that, but for the patent license, your conveying the covered work in a country, or your recipient's use of the covered work in a country, would infringe one or more identifiable patents in that country that you have reason to believe are valid.

If, pursuant to or in connection with a single transaction or arrangement, you convey, or propagate by procuring conveyance of, a covered work, and grant a patent license to some of the parties receiving the covered work authorizing them to use, propagate, modify or convey a specific copy of the covered work, then the patent license you grant is automatically extended to all recipients of the covered work and works based on it.

A patent license is “discriminatory” if it does not include within the scope of its coverage, prohibits the exercise of, or is conditioned on the non-exercise of one or more of the rights that are specifically granted under this License. You may not convey a covered work if you are a party to an arrangement with a third party that is in the business of distributing software, under which you make payment to the third party based on the extent of your activity of conveying the work, and under which the third party grants, to any of the parties who would receive the covered work from you, a discriminatory patent license (a) in connection with copies of the covered work conveyed by you (or copies made from those copies), or (b) primarily for and in connection with specific products or compilations that contain the covered work, unless you entered into that arrangement, or that patent license was granted, prior to 28 March 2007.

Nothing in this License shall be construed as excluding or limiting any implied license or other defenses to infringement that may otherwise be available to you under applicable patent law.

12. No Surrender of Others' Freedom.

If conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot convey a covered work so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not convey it at all. For example, if you agree to terms that obligate you to collect a royalty for further conveying from those to whom you convey the Program, the only way you could satisfy both those terms and this License would be to refrain entirely from conveying the Program.

13. Use with the GNU Affero General Public License.

Notwithstanding any other provision of this License, you have permission to link or combine any covered work with a work licensed under version 3 of the GNU

Affero General Public License into a single combined work, and to convey the resulting work. The terms of this License will continue to apply to the part which is the covered work, but the special requirements of the GNU Affero General Public License, section 13, concerning interaction through a network will apply to the combination as such.

14. Revised Versions of this License.

The Free Software Foundation may publish revised and/or new versions of the GNU General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies that a certain numbered version of the GNU General Public License “or any later version” applies to it, you have the option of following the terms and conditions either of that numbered version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of the GNU General Public License, you may choose any version ever published by the Free Software Foundation.

If the Program specifies that a proxy can decide which future versions of the GNU General Public License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Program.

Later license versions may give you additional or different permissions. However, no additional obligations are imposed on any author or copyright holder as a result of your choosing to follow a later version.

15. Disclaimer of Warranty.

THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM “AS IS” WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU.

SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

16. Limitation of Liability.

IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MODIFIES AND/OR CONVEYS THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

17. Interpretation of Sections 15 and 16.

If the disclaimer of warranty and limitation of liability provided above cannot be given local legal effect according to their terms, reviewing courts shall apply local law that most closely approximates an absolute waiver of all civil liability in connection with the Program, unless a warranty or assumption of liability accompanies a copy of the Program in return for a fee.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively state the exclusion of warranty; and each file should have at least the “copyright” line and a pointer to where the full notice is found.

<one line to give the program’s name and a brief idea of what it does. >

Copyright (C) <year> <name of author>

This program is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program. If not, see <<https://www.gnu.org/licenses/>>.

Also add information on how to contact you by electronic and paper mail.

If the program does terminal interaction, make it output a short notice like this when it starts in an interactive mode:

<program> Copyright (C) <year> <name of author>

This program comes with ABSOLUTELY NO WARRANTY; for details type `show w'.

This is free software, and you are welcome to redistribute it under certain conditions; type `show c' for details.

The hypothetical commands `show w' and `show c' should show the appropriate parts of the General Public License. Of course, your program's commands might be different; for a GUI interface, you would use an “about box”.

You should also get your employer (if you work as a programmer) or school, if any, to sign a “copyright disclaimer” for the program, if necessary. For more information on this, and how to apply and follow the GNU GPL, see
<<https://www.gnu.org/licenses/>>.

The GNU General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Lesser General Public License instead of this License. But first, please read
<<https://www.gnu.org/licenses/why-not-lgpl.html>>.

面向 PDF 文档的图表内容智能理解与问答系统研究与实现

摘要

随着数字化转型的加速，大量的文档资料以 PDF 格式存储。这些文档中蕴含着丰富的信息，对于大模型训练和自然语言处理领域具有重要价值。PDF 文件是中小企业常用的数字化文档文件形式。在中小企业数字化转型过程中，作为常见文件的 PDF 文档的格式较为复杂，它包含文本、图像、表格等多种元素，直接提取文本信息存在一定的难度。本论文介绍了一种可以无需较高配置就可以在本地离线运行的 PDF 智能摘要生成系统，本课题设计的系统使用开源 PDF 分析工具 npm-PDFreader 使用 DeepSeek-R1: 1.5B 的 OLLama 和 MongoDB 后端与用户对接的前后端。

对于 PDF 文件形成的长文本直接使用大模型输入复杂信息时，直接输入较长段落经常会产生幻觉造成严重的处理错误。同时 transformer 变换器模型还会出现阅读范围过小的问题：如果不加处理直接输入长文件信息，大模型权重读取机制会倾向于只阅读开头和结尾，而几乎完全无视长文件中间部分绝大多数的具体内容详细信息。因此如果未使用处理接口切片处理的情况下，直接把文件文字不加切片地输入模型，会导致不加切片直接处理生成的摘要信息出现严重遗漏缺陷。

TaPERA 接口和 AI-HPC 训练方法、ScaleFS 大文件服务器操作系统使用分级树状方式、切分不同方面问题的处理思路，为本课题的系统提供了一个处理较长 PDF 文件的启发。为完整阅读文件的全文内容生成更加准确的 PDF 内容摘要，本课题使用对长文件分段进行切片树 Slice-Tree 总结方式智能问答生成 PDF 长文件文字和表格的摘要，从而解决大模型直接读取文件时出现的大量信息遗漏问题。本课题采用的 PDF 后端处理方式将 npm-PDFreader 读取的 PDF 文件文字段落内容将较长 PDF 文件首先直接通过 npm-PDFreader 提取出文档内部的文字和表格的详细内容。随后本课题系统将树状切分的子切片自动投入 OLLama 上面的本地 Deepseek 小体量模型总结出子级总结，最后将树状子级总结合并改写为 1000 字的最终的文件摘要总结。

论文完成的主要工作有如下三部分：

- (1) 开发用户登陆界面 vue 前端。本课题采用 vue 2 前端运行本地服务，与改编

的 PDF 读取后端连接，与 Ollama 模型本地部署开源平台对接，与 node.js 服务的 MongoDB 平台后端交互；

(2) 设计完整的基于 node.js 开源平台后端。本篇论文搭建的完整 node.js 开源的本地平台后端分为五部分：node.js 主接口、本地离线路由、MongoDB 开源数据库本地离线接口、PDF 文字读取服务接口、PDF 长字符串机器学习本地处理接口。五个模块分工明确，各自高效完成 PDF 长文档摘要生成任务的一部分；

(3) 提出了一种 PDF 长字符串处理接口 Slice-Tree，用于减少本地人工智能小体量模型对长字符串处理读取范围过小的缺点。

论文所提的基于 Vue 2 前端运行本地服务与 node.js 开源平台后端、Ollama 开源平台小体量运行 Deepseek-R1 本地离线的模型后端、npm-PDFreader 开源平台后端交互、基于 node.js 开源平台后端搭建完成，对于 ai 本地部署小体量模型处理长字符串内容提高准确性具有一定的积极意义。

关键词 Deepseek PDF vue MongoDB 本地服务 智能 离线 接口

Research and Implementation of Intelligent Understanding and Question-Answering System for Chart Content in PDF Documents

Abstract

With the acceleration of digital transformation, vast quantities of document resources are being stored in PDF format. These documents contain rich information that holds significant value for large model training and natural language processing domains. PDF files serve as prevalent digital document formats, are commonly adopted by small and medium-sized enterprises (SMEs). During SME digital transformation processes, the complex structure of PDF documents - containing text, images, tables, and other heterogeneous elements - presents inherent challenges in direct text information extraction. This thesis presents an open-source local deployment solution without the need of very expensive or dedicated server, utilizing npm-PDFreader for document analysis, integrated with the DeepSeek-R1:1.5B model through open-sourced OLLama, and implemented with open-source MongoDB backend for user interaction through frontend-backend interfaces and long-text summarization process.

Direct processing of lengthy text extracted from PDF documents using large language models frequently induces hallucination, leading to critical processing error. Conventional transformer architectures suffer from limited contextual window constraints: If the user input very long text without slice-cutting process, the self-attention mechanism of large-scale models would have a tendency to read only the head and the tail of a long text information, and would almost completely ignore almost all of the information contained in the middle of a long context, resulting in serious inaccuracy for the process of long documents.

The TaPERA interface methodology and AI-HPC training method, the ScaleFS file operating system designed for professional server provides conceptual inspiration for handling extended PDF documents in our proposed system. Our system uses a Slice-Tree segmentation and summarization approach: The PDF processing backend initially extracts textual content through npm PDFreader, subsequently decomposing long documents into hierarchical segments, so that the Slice-Tree segmentation and summarization approach of

our system could solve the information-loss problem of the direct input of long text. These sub-segments would overcome automated summarization using the local DeepSeek-R1: 1.5B tiny model deployed on OLLama, with final integration of a 1,000-word comprehensive summary.

The principal work of this thesis includes the following three components:

- (1) Development of a Vue 2-based user authentication frontend interface operating, running as a local service that interacts with local-adapted PDF reading server backend, open-source OLLama local model offline running platform, and Node.js backend infrastructure communicated with MongoDB local service, .
- (2) Design of a complete Node.js-based open-source backend platform, systematically organized into five modular components: primary Node.js interface, local offline routing, MongoDB database interface, PDF text extraction backend adaptation, and Slice-Tree machine learning process interface for extended text sequences.
- (3) Proposal of a specialized PDF long-text processing interface to mitigate the limited context window constraints inherent in local AI compact models.

The implemented system architecture - featuring Vue 2 frontend services interfacing with Node.js backend infrastructure with MongoDB service, open-source platform OLLama running small-scaled local offline model Deepseek-R1 backend, and an adapted local server backend of open-source PDF-reader platform npm-PDFreader - demonstrates practical significance for enhancing processing accuracy when employing locally deployed compact AI models to handle extended textual content.

KEY WORDS DeepSeek PDF Vue MongoDB LocalService Intelligent Offline Interface

目录

第一章 绪论	1
1.1 课题的背景	1
1.2 国内外研究现状	1
1.3 论文的组织结构和本章的内容小结	2
第二章 相关理论与技术研究.....	3
2.1 Deepseek 大模型注意力运算和兼容可用性的国内外理论基础.....	3
2.1.1 中文的 ChID 俗语模型理论.....	4
2.1.2 LSTM 单层网络模型训练理论	6
2.1.3 RMS-Norm 模型理论和 Deepseek 改进的 MTP 理论.....	7
2.1.4 CQA 模型反馈结构理论	10
2.1.5 Deepseek 改进的 MLA 反馈结构模型理论	11
2.1.6 RoPE 旋转位置编码理论	13
2.1.7 Deepseek 改进的 MoE 反馈结构模型理论与强化学习	16
2.1.8 Deepseek 改进的 AI-HPC 高效率训练方法.....	18
2.2 Deepseek 与 Llama 两种开源大模型的运行环境理论.....	18
2.3 本章的内容小结	19
第三章 系统前后端的结构与实现.....	21
3.1 系统的需求分析	21
3.2 系统的概要设计	22
3.2.1 系统的模块组成	22
3.2.2 系统的领域模型	23
3.2.3 系统的操作契约	24
3.2.4 系统的时序设计	26
3.2.5 系统的开发工具	29

3.3 系统的数据结构	29
3.4 系统的后端结构与实现	32
3.4.1 本地 MongoDB 数据库访问后端	32
4.1.2 本地的 OLLama 人工智能模型开源平台后端	37
3.5 系统的前端结构与实现	38
3.5.1 系统的前端的概要设计	38
3.5.2 系统的前端配置与用户管理功能的实现	40
3.5.2 系统的前端文件处理功能实现	42
3.6 本章的内容小结	42
第四章 系统的模型处理接口 Slice-Tree 算法与 PDF 读取接口	44
4.1 本地小体量模型长文字处理切片树接口 Slice-Tree	44
4.1.1 构造切片树处理接口的需求和原因	44
4.1.2 切片树处理方法的目标和原则	44
4.1.3 切片树的构造和生成方法	45
4.2 PDF 文档格式与 PDF 文字和图表读取处理接口	49
4.2.1 PDF 文档的整体格式	49
4.2.2 PDF 文件的整体读取方法	50
4.2.3 PDF 文件对长数据流的二进制压缩存储和读取方式	52
4.2.4 PDF 文件对图片和表格数据流的存储和读取方式	54
4.2.5 以开源接口改编的 PDF 文件的文字和表格内容提取后端	55
4.3 本章的内容小结	60
第五章 系统的运行效果和测试评估	61
5.1 系统的运行效果	61
5.1.1 系统的软件运行环境和硬件配置	61
5.1.2 系统的前端运行效果	62
5.1.3 系统的后端运行效果	65
5.2 切片树接口 Slice-Tree 对系统准确性提升的效率评估	67
5.2.1 输入样例生成程序	67

5.2.2 模型的运行结果评估程序.....	69
5.2.3 接口算法性能评估实验数据与测试结论.....	70
5.3 本章的内容小结.....	75
第六章 总结.....	76
参考文献.....	77

第一章 绪论

1.1 课题的背景

随着数字化转型的加速，大量的文档资料以 PDF 格式存储。这些文档中蕴含着丰富的信息，对于大模型训练和自然语言处理领域具有重要价值。在中小企业数字化转型过程中，作为常见文件的 PDF 文档的格式复杂，包含文本、图像、表格等多种元素，直接提取文本信息存在一定的难度。所以处理和解析 PDF 文件，基于大模型等组件实现其中关于图片和表格等富文本信息理解的系统，是帮助学习专业知识和查找论文的师生、帮助中小企业低成本数字化转型升级的有力工具。

直接使用人工智能处理较长文档时会出现人工智能处理信息阅读内容过短造成人工智能信息处理不全的问题，本地部署大模型有利于处理文档时兼顾用户隐私。大模型权重读取机制会倾向于只阅读开头和结尾，而几乎完全无视长文件中间部分绝大多数的具体内容详细信息。因此如果未使用处理接口切片处理的情况下，直接把文件文字不加切片地输入模型，会导致不加切片直接处理生成的摘要信息出现严重遗漏缺陷。

本课题旨在解决无接口和非本地部署模型遇到的缺陷，制作一种基于可以使用无需高端配置，就可以本地部署的人工智能摘要系统，对 PDF 文件进行全文阅读和信息较为完备的摘要生成的本地部署的文件处理工具，用于将用户提交的 PDF 文件生成信息较完整的摘要。

1.2 国内外研究现状

DeepSeek-R1 是一款由深度求索人工智能基础技术研究有限公司开发的 2025 年发布的开源大模型，Deepseek 使用变换器（transformer）架构。DeepSeek-R1 具有 1.5B、7B、8B、14B、32B、70B 总计 6 个不同的版本^[1]，其中 1.5B 版本规模最小只有不到 2GB 数据的情况下也可以实现无需高端显卡的本地化部署并实现基础的语言逻辑。

ScaleFS 大文件服务器级操作系统使用的服务器文件分级树状处理方式，TaPERA 接口和 AI-HPC 训练方法，为本课题提供了一个切片处理思路的启发。

ScaleFS^[2]是一种面向超大规模文件提供云服务的大语言模型适配的操作系统的文件系统，它具有高性能、大批量文件负载的分布式文件操作系统存储方式设计。ScaleFS 的元数据通过键值对的形式存储在键值存储中，采用索引块和元数据块相结合的方式管理，每个元数据块存储多条记录，块依据关键字排序从而进行范围查询。启动时，它部

分元数据块和索引块会预加载至内存，随后用 LRU 策略管理缓存以确保元数据命中率。

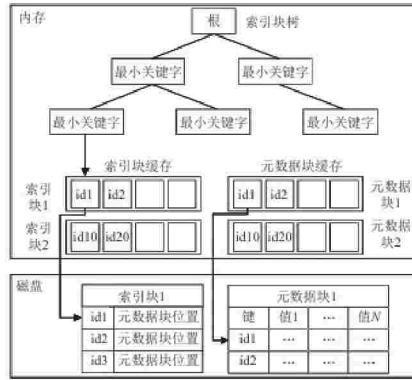


图 1-1 ScaleFS 的超大规模文件分布式操作系统文件存储方式^[12]

为了减少人工智能大语言模型幻觉的错误处理结果，Yale University 的 Yilun Zhao 提出了一种基于早期 2023 年 Meta 公司开发和发布的 Llama 开源本地模型的接口 TaPERA 使用模块化方法^[5]。TaPERA 接口将问题分解为不可细分的子问题交给大模型处理，以提高问答准确率。TaPERA 整个过程分为 3 部分：(1) 问答 QA 内容规划器将输入问题迭代分解为子问题；(2) 表执行推理器为每个子问题生成可执行的 Python 程序；3) 大模型输出答案生成器输出生成长格式答案。TaPERA 模型处理接口模块化框架更擅长表格推理，并且长格式答案准确率更高。

TaPERA 接口的 LFTQA 任务可以表述为：输入是用户查询 Q 和表 T 。表 $T = W \cup \{T_{ij} | i \leq R_T, j \leq C_T\}$ 有 R_T 行和 C_T 列，其中 W 是表标题， T_{ij} 是第 (i, j) 个单元格中的内容。生成一个段落长的答案 $Y = (y_1, y_2, \dots, y_n)$ 给定查询 Q 和源表 T ：

$$Y = \operatorname{argmax} \prod_{i=1}^n P(y_i | y < i, Q, T, \theta) \quad \text{式 (1-1)}$$

其中 θ 表示神经文本的参数生成模型， y_i 表示生成的答案， argmax 是找到给定函数或数组中最大值的索引。

1.3 论文的组织结构和本章的内容小结

本章第 1.1 节项目背景介绍了本课题项目解决的实际问题，本章第 1.2 节国内外研究现状介绍了国内外处理大模型部署问题的大型服务器文件和接口处理技术方案。

本论文分为二到六章，分别介绍本课题接口算法的理论背景、构建人工智能 PDF 摘要本地离线生成系统的系统设计和模块结构、讲解本地模型处理接口 Slice-Tree 切片树算法与 PDF 文件文字内容开源读取接口、本地离线系统测试以及有关工作的总结。

第二章 相关理论与技术研究

2.1 Deepseek 大模型注意力运算和兼容可用性的国内外理论基础

本章节介绍了以 Deepseek 和 Llama 为代表的 transformer 架构的开源大语言本地模型注意力机制学习和运算方式，分析机器学习注意力运算判定过程逻辑判断准确性产生必然的机器学习的不均等性，导致直接读取条件下的信息遗漏问题。这是因为大模型运算的不均等分析的注意力神经网络各层为了提高逻辑判断准确性，会同时导致遗漏长文本中部信息读取自身内在运算方式造成的。模型自身运算理论的提高逻辑判断准确性的必然要求需要不均衡读取，这和避免长文本读取信息遗漏的需求是矛盾的。这就是本课题所述程序在 PDF 文件读取的开源模块的基础上，设计用于本地模型读取处理长文本的开源读取接口的必要性。

在机器学习的注意力机制的运算过程中，为提高模型逻辑判断准确率，各层都会加入不均衡读取矩阵机制提高语言模型的逻辑思维能力，但是这种不均衡矩阵机制也会必然导致整体模型呈现出读取长文本的信息遗漏问题。从 RoPE 旋转矩阵底层字符编码层开始，LSTM 基础神经网络编码、ChID 语言俗语层编码、RMSNorm 归一化矩阵、MTP 多模块衔接、MoE/FFN 前向传播网络、预思考强化学习 GRPO 都会使用不均衡概率矩阵阅读，提高逻辑思维能力和判断准确性，这种不均衡读取是提高机器学习概率判断准确性的必然要求。

通过介绍 Deepseek 和 Llama 为代表的 transformer 架构模型的运算基础逻辑，本节从理论基础层面论述：由于机器学习的语言模型的注意力运算机制结构理论和运算方法必然为了逻辑判断准确性而读取运算各级都采用内在不均衡特征，本地部署的模型需要使用较长文字的读取和处理接口结构，例如本课题系统原创的切片处理的细化长文字处理接口算法 Slice-Tree，才可以应用中较为完整地读取长段落的文字信息。

本章节同时介绍了以 Deepseek 和 Llama 为代表的 transformer 架构的开源大语言本地模型部署的兼容性、可用性，论证以 Deepseek-R1 为代表的 transformer 的开源大语言模型之间具有兼容性的优点，论证以 Deepseek 和 Llama 为代表的开源大模型可以不一定必需昂贵高性能的固件，就可以在 OLLama 这种开源的平台上面离线运行的理论基础。Transformer 类模型的相通的基础理论提供的兼容性优势，为本平台不需要特别高端的硬件就可以在普通计算机上面离线运行提供了理论基础，这种兼容性优势非常适合缺乏财力和物力的普通人和中小企业普及使用离线人工智能服务，促进和帮助中小企业数字

化转型升级、促进发展成果惠及广大中小企业和更多普通人。本课题研究的系统，就是这样的一个普及性促进发展成果惠及普通人的应用。

主流大语言 transformer 模型各层整体架构为^[1]: 输入提示标记 -> 嵌入层-(可选掩码多头注意力机制) -> 层归一化 -> 可选前馈神经网络 -> 层归一化 -> N 个 transformer 层 -> 线性层 -> Softmax 输出层 -> 输出回答概率分布。例如 DeepSeek-V3 大模型的嵌入层是 RMSNorm，它的多头注意力机制为改进的 Deepseek-MLA/MTP，它的前馈神经网络 FFN 为改进的 Deepseek-MoE/FFN。每一层 Deepseek 模型的变换器执行的具体步骤为：-> RMSNorm -> Deepseek-MLA/MTP -> RMSNorm -> Deepseek-MoE/FFN ->。DeepSeek-R1 的微观设计^[2]改进和参考了 Llama 的 Pre-Norm、RMSNorm、SwiGLU 激活函数和分组注意力机制 GQA^[14]，而相对地次要地使用多头机制 MoE^[15]。

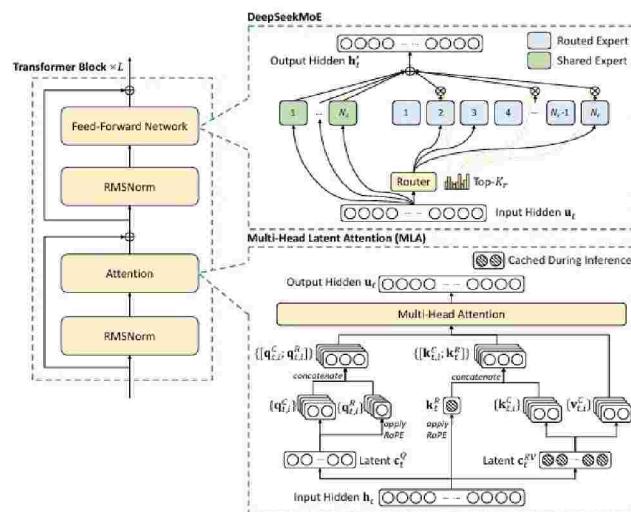


图 2-1 Deepseek 本地部署大模型结构^[3]

2.1.1 中文的 ChID 俗语模型理论

Deepseek 对中文编码使用 ChID 使用 LSTM 算法^[16]的俗语统计词汇表^[8]。ChID 统计常见语句俗语生成一个俗语库 c_i ，将语句里面的俗语以俗语嵌入向量库 c_i 的序号表示代替俗语的文字。中文的高效率语言模型 ChID 的计算方法是：

$$\alpha_i = \text{softmax}\left(c_i(\text{LSTM}(W_{1b}) \oplus \text{LSTM}(W_b|p|)^T)\right) \quad \text{式 (2-1)}$$

其中 $|p|$ 记作长度； W_{1b} 和 $W_b|p|$ 记作输入的语句，圈加号 \oplus 代表连接字符串， α_i 代表最有可能的答案， c_i 表示候选俗语的嵌入向量，对于一个输入向量 $z=[z_1, z_2, \dots, z_K]$ ，Softmax 函数将其转换为概率分布 $p=[p_1, p_2, \dots, p_K]$ ，每个元素 p_i 为：

$$\text{softmax}(p) = \frac{e^{z_i}}{\sum_{j=1}^k e^{z_j}} \quad \text{式 (2-2)}$$

中文的 ChID 读取方式为：斯坦福注意力阅读器（Stanford Attentive Reader, SAR）：SAR 首先进行双线性注意力计算^[29]：使用双线性矩阵 W_s 计算注意力权重，生成上下文向量 o ：

$$s_t = \text{softmax}_t(h_b^T W_s h_t) \quad \text{式 (2-3)}$$

$$o = \left(\sum_{t=1}^{|p|} s_t h_t \right) \quad \text{式 (2-4)}$$

SAR 随后进行候选成语评分：直接通过上下文向量 o 与候选成语嵌入的相似度计算得分：

$$\alpha_i = \text{softmax}_i(o^T c_i) \quad \text{式 (2-5)}$$

中文的 ChID 的 c_i 表示候选俗语的嵌入向量计算方式如下^[7]：使用 DSG 向量跳字法(Directional Skip Gram)确定俗语的方向向量。引入方向向量 $\delta_{w_{t+i}}$ ，计算方向感知概率 $g(w_{t+i} | w_t)$ ：

$$g(w_{t+i} | w_t) = \frac{e^{\delta_{w_{t+i}}^T v_{w_t}}}{\sum_{w \in V} e^{\delta_{w_{t+i}}^T v_w}} \quad \text{式 (2-6)}$$

其中， $\delta_{w_{t+i}}$ 是上下文的词组 W_{t+i} 的方向向量，用于编码其相对中心词 W_t 的位置（左或右）。

DSG 向量跳字法参数更新优化算法：通过梯度下降联合优化词向量 v 和方向向量 δ 。对于每个上下文词 W_{t+i} ，更新规则是：中心词向量 V_{w_t} 的更新：

$$V_{w_t}(\text{new}) = V_{w_t}(\text{old}) - \gamma [\sigma(V_{w_t}^T \delta_{w_{t+i}}) - D] \delta_{w_{t+i}} \quad \text{式 (2-7)}$$

方向向量 $\delta_{w_{t+i}}$ 的更新：

$$\delta_{w_{t+i}}(\text{new}) = \delta_{w_{t+i}}(\text{old}) - \gamma [\sigma(V_{w_t}^T \delta_{w_{t+i}}) - D] V_{w_t} \quad \text{式 (2-8)}$$

其中 σ 是 Sigmoid 函数把内积到概率区间[0,1]的映射。D 是方向标签：若 W_{t+i} 在左侧 ($i < 0$)，则 $D=1$ ；若 W_{t+i} 在右侧 ($i > 0$)，则 $D=0$ 。 γ 为学习率，控制参数更新步长。

SwiGLU 是^[13]一种激活函数：前向反馈神经网络

$$\text{FFN}(x, W_1, W_2, b_1, b_2) = \max(0, x W_1 + b_1) W_2 + b_2 \quad \text{式 (2 - 9)}$$

$$\text{FFNSwish}(x, W_1, W_2, b_1, b_2) = \text{Swish}_1(x W_1) W_2 \quad \text{式 (2 - 10)}$$

$$\text{SwiGLU}(x, W, V, b, c, \beta) = \text{Swish}_\beta(x W + b) \otimes (xV + c) \quad \text{式 (2 - 11)}$$

其中 \otimes 是向量逐元素相乘的 Hadamard 积（符号 ‘ \otimes ’ 在此不是外积）^[10]。

$$\text{FFNSwiGLU}(x, W, V, W_2) = (\text{Swish}_1(x W) \otimes xV) W_2 \quad \text{式 (2 - 12)}$$

Swish 定义为：

$$\text{Swish}(x) = x (\text{sigmoid}(Bx)) \quad \text{式 (2 - 13)}$$

$$\text{sigmoid}(z) = \frac{1}{1 + e^{-z}} \quad \text{式 (2 - 14)}$$

其中 B 为常量参数。

2.1.2 LSTM 单层网络模型训练理论

LSTM 是一种 1997 年由当时在慕尼黑工业大学的 Sepp Hochreiter 等提出的单层 transformer 变换器模型训练算法^[16]。设 LSTM 输入为 x_t （当前层输入）， h_{t-1} （上一层状态）， C_{t-1} （上一层细胞状态）^[17]，LSTM 依次通过遗忘门、输入门、更新态和输出门^[16]四个步骤完成单层神经网络的计算：

(1) LSTM 遗忘门的输出是 f_t ， f_t 取值 0~1，控制细胞状态的遗忘比例^[22]。设 W_f 、 b_f 是遗忘门的权重矩阵和偏置； σ 是 Sigmoid 函数将取值范围减小为 [0,1]，则输出：

$$f_t = \sigma(W_f[h_{t-1}, x_t] + b_f) \quad \text{式(2-15)}$$

(2) LSTM 输入门的输出 i_t 取值 0~1，控制候选值的保留比例。设 \tilde{C}_t 是通过 \tanh 函数生成新信息的候选细胞状态，则：

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \quad \text{式(2-16)}$$

$$\tilde{C}_t = \tanh(W_c \cdot [h_{t-1}, x_t] + b_c) \quad \text{式(2-17)}$$

(3) LSTM 更新态细胞状态 C_t 是对遗忘门过滤的旧信息和输入门筛选的新信息进行综合运算。设 \odot 是逐元素相乘 (Hadamard 积)：

$$C_t = f_t \odot C_{t-1} + i_t \odot \tilde{C}_t \quad \text{式(2-18)}$$

(4) LSTM 输出门 (Output Gate) 如下：设 o_t 是输出门控制的当前隐藏状态的生成比例， h_t 是当前时刻的用于传到下一时刻或输出的隐藏状态，则：

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \quad \text{式(2-19)}$$

$$h_t = o_t \odot \tanh(C_t) \quad \text{式(2-20)}$$

2.1.3 RMSNorm 模型理论和 Deepseek 改进的 MTP 理论

RMSNorm 是一种 transformer 变换器模型前置规范化（也就是层归一化）算法^[9]，RMSNorm 通过计算多语言相邻词语频率确定成语和词语并增加区分概率准确性的训练方式。设输入向量 $x \in \mathbb{R}^m$ ，FFN 前向传播神经网络输出向量 $y \in \mathbb{R}^n$ ，通过一个线性变换和非线性激活如下：

$$a_i = \sum_{j=1}^m w_{ij} x_j \quad \text{式(2-21)}$$

$$y_i = f(a_i + b_i) \quad \text{式(2-22)}$$

其中 w_i 表示第 i 个输出神经元, b_i 表示偏移量(bias scalar)初始值为 0, $f()$ 是一个非线性一一对应激活函数, $a \in R^n$ 记作节点的输入权重(weight-summed input), a 是规范化的目标。RMSNorm 均方根规范化使用这种方法进行规范化:

$$\bar{a}_i = \frac{(a_i g_i)}{\text{RMS}(a)} \quad \text{式(2-23)}$$

$$\text{RMS}(a) = \sqrt{\frac{1}{n} \sum_{i=1}^n a_i^2} \quad \text{式(2-24)}$$

其中 $g_i \in R^n$ 是用于重新改变输入规模的生成参数(gain parameter), 初始值设为 1。

Deepseek-MTP 首先在主模型输入 token 后经过目标词汇 token 的交叉熵损失(Cross-Entropy Loss)以后进入下一个 MTP 模块, 通过 RMSNorm 和另一个通过 RMSNorm 的输入词汇 token 合并为线性投影, 再经过主模型 Transformer 块经过目标词汇 token 的交叉熵损失函数模块计算(Cross-Entropy Loss)以后进入下一个 MTP 模块。MTP 使用 D 个词汇序列模组进行训练^[3], 预测 D 个后面的词汇 token, 第 k 层 MTP 模组包括一个共享的嵌入层 $\text{Emb}(\cdot)$, 共享的输出函数层 $\text{OutHead}(\cdot)$, 变换器主模型模块 $\text{TRM}_k(\cdot)$, 投影矩阵(projection matrix) $M_k \in R^{d \times 2d}$, 对第 i 个输入词汇 t_i , 在第 k 个预测深度, 混合第 $k-1$ 层的深度 $h_i^{k-1} \in R^d$ 是第 $i+k$ 个词汇嵌入 $\text{Emb}(t_{i+k}) \in R^d$, 线性投影:

$$h'_i^k = M_k [\text{RMSNorm}(h_i^{k-1}); \text{RMSNorm}(\text{Emb}(t_{i+k}))] \quad \text{式(2-25)}$$

其中 $[\cdot; \cdot]$ 表示拼接, 当 $k=1$ 特例表示原有 RMSNorm 模型。

每一个 MTP 模块嵌入层与主模型共用, 线性的投影 h'_i^k 作为 transformer 变换器模块在第 k 层 的输入以生成当前深度 h_i^k :

$$h_{1:T-k}^k = \text{TRM}_k (h'_{1:T-k}^k) \quad \text{式(2-26)}$$

其中 T 表示输入序列长度, d_{ij} 记作切片(slicing)运算也就是数列/矩阵取左右边界(Boundaries)。最后以 h_i^k 为输入值, 共享的输出头计算第 k 个额外预测的概率分布

$P_{i+1+k}^k \in R^V$, 其中 V 是词汇表大小。

$$P_{i+1+k}^k = \text{Outhead}(h_i^k) \quad \text{式(2-27)}$$

其中输出使用 $\text{Outhead}(\cdot)$ 输出函数, 线性映射和代表归一化(logit), 随后使用 $\text{Softmax}(\cdot)$ 函数运算第 k 个额外 token 词汇的概率。

其中 logit 函数:

$$\text{logit}(L) = \lg\left(\frac{L}{1-L}\right) \quad \text{式(2-28)}$$

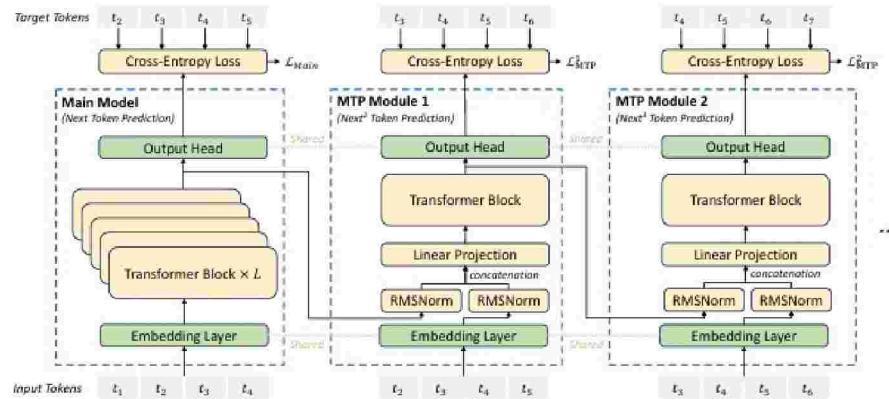


图 2-2 Deepseek 的 MTP 强化学习^[3]

MTP 的交叉熵损失模块^[30] (Cross-Entropy Loss)为:

$$L_{MTP}^k = \text{CrossEntropy}(P_{2+k:T+1}^k, t_{2+k:T+1}) = -\frac{1}{T} \sum_{i=2+k}^{T+1} \log P_i^k[t_i] \quad \text{式(2-29)}$$

其中 T 记作输入序列的长度, t_i 记作 i 位置上接地真词汇 ground-truth token, $P_i^k[t_i]$ 记作概率 t_i 的对应预测概率并属于第 k 个 MTP 模块。最终计算全部 MTP 损失的平均数, 并乘以权重 G , 得到 MTP 损失函数:

$$L_{MTP} = \frac{G}{D} \sum_{k=1}^D L_{MTP}^k \quad \text{式(2-30)}$$

2.1.4 CQA 模型反馈结构理论

GQA^[14]是一种将输出层反馈于输入的自反馈训练方式。GQA 的训练将询问头分为 G 组，每组共享一个键头 (key head) 和值头(value head)。CQA-G 代表 G 组分组查询；CQA-I 是单组分组查询具有单独的键头和对应的值头与多头机制 MoE^[15]等价。多头机制 MoE 状态下键 key-值 value-查询 query 一一对应；CQA 状态下一组键 key-值 value 对应多个查询 query。

GQA 的基本目标是^[25]输入自然语言问题 q，知识图谱 G，主题实体 e_{te} ，约束实体集 E_c ，推理路径长度 L；输出决策路径 h、和节点约束 c^* 。GQA 的基础步骤是：

1、对 L 层的每个当前状态 s_t ，计算其动作空间 $A(s_t)$ ：

$$A(s_t) = \{r \mid (e_t, r) \in G\} \quad \text{式 (2-31)}$$

2、对动作空间 $A(s_t)$ 内每一个动作 a，计算动作 a 与问题 q 的语义分数 $S(a, q)$ 算出概率分布 $P(a \mid s_t)$ 。语义分数 $S(a, q)$ 和概率分布 $P(a \mid s_t)$ 的计算方式是：

$$S(a, q) = r W_2 q_1'^T \quad \text{式 (2-32)}$$

$$q_1'^T = G \otimes q_1 \quad \text{式 (2-33)}$$

$$G = \text{softmax}(W_1 \cos(r \times q_1^T)) \quad \text{式 (2-34)}$$

$$P(a \mid s_t) = \text{softmax}(S(a, q)) \quad \text{式 (2-35)}$$

其中， q_1 是自然语言提问向量； G 是中介权重参数向量； W_1 是模型词汇权重参数向量； r 是参数空间 $A(s_t)$ 内 a 在 t 的每个取值； $a \otimes b$ 表示两个向量 a、b 逐元素乘积 (Hadamard 积)。

3、根据每个 a 的动作选择概率 $P(a \mid s_t)$ 求合并参数空间内计算实体选择概率 E，并根据实体选择概率 E 最大值路径确定和延长决策路径 h，根据实体选择概率 E 确定每个

节点的节点约束 c^* 。当出现路径歧义时，计算比较不同路径 h 与问题 q 的向量夹角 \cos 值，选择和问题向量夹角更小的路径 h 作为最终 GQA 选择路径。

2.1.5 Deepseek 改进的 MLA 反馈结构模型理论

Deepseek-MLA^[3] 则是 CQA 注意力机制的一种改进。设 d 表示嵌入维度， n 表示注意力头的数量， d_h 表示每个头的维度， $h_t \in R_d$ 表示第 t 个词元在给定注意力层的输入。MLA 的核心是通过对注意力键（Key）和值（Value）进行低秩联合压缩(low-rank joint compression)，以减少推理过程中的键值（KV）缓存。

Deepseek-MLA 联合压缩键值潜在向量：

$$C_t^{KV} = W^{DKV} h_t \quad \text{式(2-36)}$$

Deepseek-MLA 键的上投影：

$$k_{t,1}^C; k_{t,2}^C; \dots; k_{t,n_h}^C = k_t^C = W^{UK} C_t^{KV} \quad \text{式(2-37)}$$

使用 RoPE 解耦键值^[19]：

$$k_t^R = \text{RoPE}(W^{KR} h_t) \quad \text{式(2-38)}$$

合并压缩键和解耦键：

$$k_{t,i} = [k_{t,i}^C; k_{t,i}^R] \quad \text{式(2-39)}$$

值的上投影是：

$$[v_{t,1}^C; v_{t,2}^C; \dots; v_{t,n_h}^C] = v_t^C = W^{UV} C_t^{KV} \quad \text{式(2-40)}$$

其中 $C_t^{KV} \in R^{d_c}$ 是键（Key）和值（Value）的潜在压缩向量； $d_c (\ll d_h n_h)$ 表示 KV 压缩维度远小于原始维度 $d_h n_h$ ； $W^{DKV} \in R^{d_c \times d}$ 是降维矩阵（Down-projection matrix）； $W^{UK}, W^{UV} \in R^{d_h n_h \times d_c}$ 分别为键和值的上投影矩阵（Up-projection matrices）；

$W^{KR} \in R^{d_h^R \times d}$ 是生成带旋转位置编码 (RoPE) 的解耦键的投影矩阵； $\text{RoPE}(\cdot)$ 表示应用旋转位置编码矩阵的操作； $[;]$ 表示向量拼接。注意 C_t^{KV} 和 k_t^R 应该在生成过程中缓存，当维持表现和标准 MHA 对比时可以显著缩小 KV 缓存。

对注意力查询，Deepseek-MLA 使用一个低级别压缩降低训练内存占用：查询的潜在向量：

$$C_t^Q = W^{DQ} h_t \quad \text{式}(2-41)$$

查询的上投影：

$$[q_{t,1}^C; q_{t,2}^C; \dots; q_{t,n_h}^C] = q_t^C = W^{UQ} C_t^Q \quad \text{式}(2-42)$$

使用 RoPE 进行解耦查询：

$$[q_{t,1}^R; q_{t,2}^R; \dots; q_{t,n_h}^R] = q_t^R = \text{RoPE}(W^{QR} C_t^Q) \quad \text{式}(2-43)$$

合并压缩查询和解耦查询：

$$q_{t,i} = [q_{t,i}^C; q_{t,i}^R] \quad \text{式}(2-44)$$

其中 $C_t^Q \in R^{d'_c}$ 是查询 (Query) 的潜在压缩向量； d'_c ($\ll d_h n_h$) 表示查询压缩维度 (远小于标准查询维度 $d_h n_h$)； $W^{DQ} \in R^{d'_c \times d}$ 是查询的降维矩阵 (Down-projection)； $W^{UQ} \in R^{d_h n_h \times d'_c}$ 是查询的上投影矩阵 (Up-projection)； $W^{QR} \in R^{d_h^R n_h \times d'_c}$ 是生成带旋转位置编码 (RoPE) 的解耦查询的投影矩阵。最后注意力查询 ($q_{t,i}$)、键 ($k_{j,i}$) 和值 ($v_{j,i}$) 通过以下方式组合，生成最终的注意力输出 u_t ：

$$o_{t,i} = \sum_{j=1}^t \text{Softmax}_j \left(\frac{q_{t,i}^T k_{j,i}}{\sqrt{d_h + d_h^R}} \right) v_{j,i}^C \quad \text{式}(2-45)$$

$$u_t = W^0 [o_{t,1}; o_{t,2}; \dots; o_{t,n_h}] \quad \text{式(2-46)}$$

其中 $W^0 \in R^{d \times d_h n_h}$ 表示输出投影矩阵。

2.1.6 RoPE 旋转位置编码理论

RoPE 旋转位置编码是一种把输入的语句生成唯一的相对高维空间向量进行编码的编码方式^[19]。令 $S_N = \{w_i\}_{i=1}^N$ 是 N 个字符串的输入语句， w_i 是它的第 i 个元素； S_N 对应的嵌入层（Embedding）记作 $E_N = \{x_i\}_{i=1}^N$ ，其中 $x_i \in R^d$ 是 token 词汇 w_i 的无位置信息的 d 维嵌入向量，自注意机制分为这 3 个定义式：

$$q_m = f_q(x_m, m) \quad \text{式(2-47)}$$

$$k_n = f_k(x_n, n) \quad \text{式(2-48)}$$

$$v_n = f_v(x_n, n) \quad \text{式(2-49)}$$

其中 q_m 、 k_n 、 v_n 是 f_q 、 f_k 、 f_v 的第 m 和 n 个位置。在常规变换器（transformer）模型里面的注意力权重 $a_{m,n}$ 、目标向量 $o_{m,n}$ 是：

$$a_{m,n} = \frac{e^{\frac{q_m^T k_n}{\sqrt{d}}}}{\sum_{j=1}^N e^{\frac{q_m^T k_j}{\sqrt{d}}}} \quad \text{式(2-50)}$$

$$o_{m,n} = \sum_{j=1}^N a_{m,n} v_n \quad \text{式(2-51)}$$

绝对位置编码理论（Absolute position embedding）是一种将语句一一对应于空间内唯一的绝对位置的语句向量生成理论，它的一般形式为：

$$f_{t:t \in \{q,k,v\}}(x_i, i) := W_{t:t \in \{q,k,v\}}(x_i + p_i) \quad \text{式(2-52)}$$

其中 $p_i \in \mathbb{R}^d$ 是取决于字词 x_i 的 d 维度向量。有一种典型三角绝对位置编码：可训练向量属于空间^[26]集合 $p_i \in \{p_t\}_{t=1}^L$, 其中 L 为最大允许的语句长度, 它的编码方式为:

$$p_{i,2t} = \sin\left(\frac{i}{10000^{\frac{2t}{d}}}\right) \quad \text{式}(2-53)$$

$$p_{i,2t+1} = \cos\left(\frac{i}{10000^{\frac{2t}{d}}}\right) \quad \text{式}(2-54)$$

其中 $p_{i,2t}$ 是 d 维度向量 p_i 的第 $2t$ 个元素。

相对位置编码理论 (Relative position embedding) 是一种使用相对位置而不需要唯一绝对位置的语句向量生成理论^[23], 它的一般形式为:

$$f_q(x_m) = W_q x_m \quad \text{式}(2-55)$$

$$f_k(x_n, n) = W_k(x_n + \tilde{p}_r^k) \quad \text{式}(2-56)$$

$$f_v(x_n, n) = W_v(x_n + \tilde{p}_r^v) \quad \text{式}(2-57)$$

其中 $\tilde{p}_r^k, \tilde{p}_r^v \in \mathbb{R}^d$ 是可训练的相对位置编码,

$$r = \text{clip}(m - n, r_{\min}, r_{\max}) \quad \text{式}(2-58)$$

r 表示位置 m 和 n 之间的相对距离, $\text{clip}(a,b,c)$ 函数将数组 a 中的元素限制在指定的最小值 b 和最大值 c 之间。

权重的决定参数 $q_m^T k_n$ 展开式^[24]:

$$q_m^T k_n = x_m^T W_q^T W_k x_n + x_m^T W_q^T W_k p_n + p_m^T W_q^T W_k x_n + p_m^T W_q^T W_k p_n \quad \text{式 }(2-59)$$

将相对位置向量 \tilde{p}_{m-n} 代替原来的绝对位置编码 \tilde{p}_m , 用两个三四象限的向量 u 和 v 与绝对位置无关, W_k 分为基于内容区分的 x_i 和基于位置区分的 p_i 编码, 记作 W_k 和 \tilde{W}_k :

$$q_m^T k_n = x_m^T W_q^T W_k x_n + x_m^T W_q^T \tilde{W}_k \tilde{p}_{m-n} + u^T W_q^T W_k x_n + v^T W_q^T \tilde{W}_k \tilde{p}_{m-n} \quad \text{式 (2-60)}$$

最后将绝对位置编码 p_m 和 p_n 完全地取代为相对位置编码 \tilde{p}_{m-n} :

$$q_m^T k_n = x_m^T W_q^T W_k x_n + x_m^T W_q^T W_k \tilde{p}_{m-n} + \tilde{p}_{m-n}^T W_q^T W_k x_n \quad \text{式 (2-61)}$$

可见 $q_m^T k_n$ 是一种不同位置等价性知识转化的参数, q_m 和 k_n 的内积(一般矩阵乘法)记为函数 g , 只需要单词嵌入 x_m 和 x_n , 它们相关的 $m-n$ 是输入变量, 要旋转位置编码 RoPE 需要的目标函数是:

$$\langle f_q(x_m, m), f_k(x_n, n) \rangle = g(x_m, x_n, m - n) \quad \text{式 (2-62)}$$

设维度 $d=2$ 情况下: 前面的 RoPE 目标函数的解为:

$$f_q(x_m, m) = (W_q x_m) e^{im\theta} \quad \text{式 (2-63)}$$

$$f_k(x_n, n) = (W_k x_n) e^{in\theta} \quad \text{式 (2-64)}$$

$$g(x_m, x_n, m - n) = \operatorname{Re}[(W_q x_m)(W_k x_n)^* e^{i(m-n)\theta}] \quad \text{式 (2-65)}$$

其中 $\operatorname{Re}[\cdot]$ 表示实数部分, 复数 $(W_k x_n)^*$ 表示 $(W_k x_n)$ 的共轭复数 (conjugate complex number), $\theta \in \mathbb{R}$ 是一个预设的非 0 常数。 $f_{\{q,k\}}$ 还可以进一步表示为矩阵乘积形式:

$$f_{\{q,k\}}(x_m, m) = \begin{pmatrix} \cos m\theta & -\sin m\theta \\ \sin m\theta & \cos m\theta \end{pmatrix} \begin{pmatrix} W_{\{q,k\}}^{(11)} & W_{\{q,k\}}^{(12)} \\ W_{\{q,k\}}^{(21)} & W_{\{q,k\}}^{(22)} \end{pmatrix} \begin{pmatrix} x_m^1 \\ x_m^2 \end{pmatrix} \quad \text{式 (2-66)}$$

其中: $\begin{pmatrix} x_m^1 \\ x_m^2 \end{pmatrix}$ 是 x_m 的 2D 矩阵形式。因此 g 可以被看做符合 RoPE 目标函数的 2D 矩阵解, 这个解的几何意义是旋转该向量 θ 角度, 因此这种编码方法被称为旋转位置编码理论 (Rotary Position Embedding)。将旋转位置编码推广到 $x_i \in \mathbb{R}^d$ 的 d 维度高维空间内, 将 d 维度空间拆分为 $d/2$ 维度空间, 将内积 $f_{\{q,k\}}$ 转为:

$$f_{\{q,k\}}(x_m, m) = R_{\Theta, m}^d W_{\{q,k\}} x_m \quad \text{式 (2-67)}$$

其中:

$$R_{\Theta,m}^d = \begin{pmatrix} \cos m\theta_1 & -\sin m\theta_1 & 0 & 0 & & 0 & 0 \\ \sin m\theta_1 & \cos m\theta_1 & 0 & 0 & & 0 & 0 \\ 0 & 0 & \cos m\theta_2 & -\sin m\theta_2 & \dots & 0 & 0 \\ 0 & 0 & -\sin m\theta_2 & \cos m\theta_2 & & 0 & 0 \\ \vdots & & & & \ddots & & \vdots \\ 0 & 0 & 0 & 0 & \dots & \cos m\theta_{d/2} & -\sin m\theta_{d/2} \\ 0 & 0 & 0 & 0 & \dots & \sin m\theta_{d/2} & \cos m\theta_{d/2} \end{pmatrix}$$

式 (2-68)

$R_{\Theta,m}^d$ 的预定义参数 Θ 的旋转矩阵。

$$\Theta = \{\theta_i = 10000^{-2(i-1)/d}, i \in [1, 2, \dots, d/2]\} \quad \text{式 (2-69)}$$

根据先前的注意力权重公式 2-56:

$$q_m^T k_n = (R_{\Theta,m}^d W_q x_m)^T (R_{\Theta,n}^d W_k x_n) = x^T W_q R_{\Theta,n-m}^d W_k x_n \quad \text{式 (2-70)}$$

$$R_{\Theta,n-m}^d = (R_{\Theta,m}^d)^T (R_{\Theta,n}^d) \quad \text{式 (2-71)}$$

其中 R_{Θ}^d 是正交矩阵 (orthogonal matrix) 以保障编码位置信息的稳定性。

2.1.7 Deepseek 改进的 MoE 反馈结构模型理论与强化学习

MoE^[3] 是另一种将输出层反馈于输入的自反馈训练方式。MoE 将输入分发给多个专家 (MLP) 处理，并通过路由机制动态选择相关专家。

Deepseek-MoE-FFN 前向传播网络将部分神经模拟“专家”隔离为共享网络模拟专家如图 2-1。令 u_t 表示第 t 个 token 的前向传播网络 (FFN) 输入，其 FFN 输出 h_t 的计算方式是：

$$h_t = u_t + \sum_{i=1}^{N_s} FFN_i^{(s)}(u_t) + \sum_{i=1}^{N_t} g_{i,t} FFN_i^{(r)}(u_t) \quad \text{式 (2-72)}$$

$$g_{i,t} = \frac{g'_{i,t}}{\sum_{j=1}^{N_r} g'_{j,t}} \quad \text{式 (2-73)}$$

$$g'_{i,t} = \begin{cases} s_{i,t}, & s_{i,t} \in \text{Topk}(\{s_{j,t} | 1 \leq j \leq N_r\}, K_r) \\ 0, & \text{其他的条件} \end{cases} \quad \text{式 (2-74)}$$

$$s_{i,t} = \text{Sigmoid}(u_t^T e_i) \quad \text{式 (2-75)}$$

其中 N_s 和 N_r 分别表示共享专家和路由专家的数量； $\text{FFN}_i^{(s)}(\cdot)$ 和 $\text{FFN}_i^{(r)}(\cdot)$ 分别表示第 i 个共享专家和第 i 个路由专家； K_r 表示激活的路由专家数量； $g_{i,t}$ 为第 i 个专家的门控值； $s_{i,t}$ 为 token 到专家的亲和度分数； e_i 是第 i 个路由专家的中心向量； $\text{Topk}(\cdot, K)$ 表示从第 t 个 token 与所有路由专家的亲和度分数中选取前 K 个最高分。

Deepseek-R1 使用强化学习方式提高准确性^[2]，这种改进的强化学习方法叫做分组相关对策优化（GRPO， Group Relative Policy Optimization）。对每个问题 q ， GRPO 会从旧对策里面 $\pi_{\theta_{\text{old}}}$ 取样一组输出 $\{o_1, o_2, \dots, o_G\}$ ，并通过以下方法求解最大化模型 π_θ ：

$$\begin{aligned} J_{\text{GRPO}}(\theta) \\ = E[q \sim P(Q), \{o_i\}_{i=1}^G \sim \pi_{\theta_{\text{old}}}(o|q)] \left(\frac{1}{G} \right) \sum_{i=1}^G \left(\min \left(\frac{A_i(\pi_\theta(o_i|q))}{\pi_{\theta_{\text{old}}}(o_i|q)}, A_i \text{clip} \left(\frac{\pi_\theta(o_i|q)}{\pi_{\theta_{\text{old}}}(o_i|q)}, 1 - \varepsilon, 1 + \varepsilon \right) \right) - \beta D_{\text{KL}}(\pi_\theta || \pi_{\text{ref}}) \right) \end{aligned} \quad \text{式 (2-76)}$$

$$D_{\text{KL}}(\pi_\theta || \pi_{\text{ref}}) = \frac{\pi_{\text{ref}}(o_i|q)}{\pi_{\theta_{\text{old}}}(o_i|q)} - \lg \left(\frac{\pi_{\text{ref}}(o_i|q)}{\pi_{\theta_{\text{old}}}(o_i|q)} \right) - 1 \quad \text{式 (2-77)}$$

其中 ε 和 β 是超参数（hyper-parameters）也就是事先用户设定的参数， A_i 是回报 $\{r_1, r_2, \dots, r_G\}$ 的优势数组，

$$A_i = \frac{r_i - \text{mean}(\{r_1, r_2, \dots, r_G\})}{\text{std}(\{r_1, r_2, \dots, r_G\})} \quad \text{式 (2-78)}$$

其中 mean 表示数组 $\{r_1, r_2, \dots, r_G\}$ 平均数， std 表示数组 $\{r_1, r_2, \dots, r_G\}$ 的标准差（也就是方差的算术平方根）。R1 模型运行时通过强化学习产生完善用户输入的提示词形成提示语句，再将完善的提示语句返回成输入生成最终答案，以提高模型的准确性。

2.1.8 Deepseek 改进的 AI-HPC 高效率训练方法

Deepseek 使用改进的 AI-HPC 训练方法进行分解式训练，将训练器分为分为 3 部分 CPU 汇总跨节点（inter-node allreduce）运算、子级 CPU 汇总子级 GPU 进行的节点间（intra-node）运算^[18]。第 1 步先进行节点间降低计算量，第 2 步使用 CPU 进行跨节点运算量分解，第 3 步使用 GPU 传输降低计算量的数据。节点间降低运算量（intra-node）的算法步骤如表 2-1 左栏，跨节点（inter-node）运算量分解算法步骤如表 2-1 右栏^[18]。

表 2-1 Deepseek 模型使用 AI-HPC 的节点间降低运算量和跨节点降低运算量方法

<p>Deepseek 模型使用 AI-HPC 的节点间(intra-node)降低运算量算法 1:</p> <pre> Data: Dg: data need to all reduce Result: Dc: data reduced in this node Split Dg by Chunk Size for Dg_i in Splited-Dg do Dc_i = MemCopyAsync Dg_i to CPU Mmeory //Transfer GPU Memory Data to CPU memory end for i in Splited_Count do while every GPU's Dg_i → Dc_i finished do //Wait for chunk-i transfer finished in this node for j in GPU Count do //do intra-node reduce Dc_i += GPU-j's Dc_i end do internode reduce(Dc_i) //do inter-node reduce end end </pre>	<p>Deepseek 模型使用 AI-HPC 的跨节点(inter-node)降低运算量算法 2:</p> <pre> Data: DL:local node reduced data by Algorithm 1 Data: DR: received other node reduced data Result: Dg: reduced data transfer to GPU for i in Chunk Size do //reduce data receive data DR_i from prev node if DL_i ready then DL_i += DR_i // reduce data end if Thread is root of Tree then send DL_i to prev node // Dg_i OK, go pass 2 else send DL_i to next node end receive data DR_i from next node //Pass 2: gather reduced data, individual thread for j in GPU Count do Dg_i = MemCopyAsync DR_i to GPU-j end // Dg_i is all reduced send DL_i to prev node </pre>
---	--

2. 2 Deepseek 与 Llama 两种开源大模型的运行环境理论

DeepSeek-R1-67B 模型宏观设计采用分组查询注意力（GQA）增加网络深度（95 层）同时扩展 FFN 宽度。DeepSeek-R1 使用超参数优化器 AdamW，其 $\beta_1=0.9$, $\beta_2=0.95$ ，权重衰减为 0.1；使用多步非余弦调度进行学习率调度，在 80% 训练词元后降至最大值的 31.6%，90% 后降至 10%，其梯度裁剪为 1.0。Deepseek 模型运行过程包括：词汇关键词确认初始化 embed 为平行分层 ParallelEmbedding(args.vocab_size, args.dim)，词汇分层 self.layers，词汇概率矩阵输出 self.norm。

表 2-2 Deepseek 模型运行的基础结构

```

def __init__(self, args: ModelArgs):
    """Initializes the Transformer model.
    Args: args (ModelArgs): Model arguments containing transformer parameters."""
    global world_size, rank
    world_size = dist.get_world_size() if dist.is_initialized() else 1
    rank = dist.get_rank() if dist.is_initialized() else 0
    Linear.dtype = torch.float8_e4m3fn if args.dtype == "fp8" else torch.bfloat16
    super().__init__()
    self.max_seq_len = args.max_seq_len
    self.embed = ParallelEmbedding(args.vocab_size, args.dim)
    self.layers = torch.nn.ModuleList()
    for layer_id in range(args.n_layers):
        self.layers.append(Block(layer_id, args))
    self.norm = RMSNorm(args.dim)
    self.head = ColumnParallelLinear(args.dim, args.vocab_size, dtype=torch.get_default_dtype())
    self.register_buffer("freqs_cis", precompute_freqs_cis(args), persistent=False)

```

Llama^[4]作为一种早期算法相对朴素的另一种开源变换器模型（transformer）可以帮助我们理解 Deepseek 模型的计算和运行方式。Llama 是总部位于加利福尼亚州的 Meta 公司开发的一种早在 2023 年发布的更早的开源大规模语言本地模型，Llama 有 7B、13B、33B、65B 这 4 种不同规模的版本，Llama 模型代表的变换器模型运行工作时的几个基本部分如图 2-3：Llama 词汇的确认初始化 idx 为 self.padding_idx，词汇分层 self.layers，词汇概率矩阵输出 self.norm。因此，为 Llama 早期的开源 transformer 模型设计的一种开源多平台本地运行模型开源平台软件 OLLama 也可以采用类似于更早开源大模型 Llama 的结构在本地服务运行 Deepseek 大模型。

```

490     class LlamaModel(LlamaPreTrainedModel):
491         """Transformer decoder consisting of *config.num_hidden_layers* layers.
492         Each layer is a [`LlamaDecoderLayer`] Args: config: LlamaConfig"""
493         def __init__(self, config: LlamaConfig):
494             super().__init__(config)
495             self.padding_idx = config.pad_token_id
496             self.vocab_size = config.vocab_size
497             self.embed_tokens = nn.Embedding(config.vocab_size, config.hidden_size, self.padding_idx)
498             self.layers = nn.ModuleList([
499                 [LlamaDecoderLayer(config, layer_idx) for layer_idx in range(config.num_hidden_layers)]
500             ])
501             self.norm = LlamaRMSNorm(config.hidden_size, eps=config.rms_norm_eps)
502             self.rotary_emb = LlamaRotaryEmbedding(config=config)
503             self.gradient_checkpointing = False
504             # Initialize weights and apply final processing
505             self.post_init()

```

图 2-3 2023 年发布的 Llama 开源大模型运行的基础结构

2.3 本章的内容小结

本章节介绍了本课题所述系统及其主要算法切片树 Slice-Tree 的原则、分块处理长

文本必要性的理论层面的需求。本章节从理论机器学习要求逻辑判断准确性的自身算法特征必然产生读取不均衡性理论基础，还介绍了以 Deepseek 和 Llama 为代表的 transformer 架构的开源大语言本地模型在自身部署的运行结构层面具有可用性兼容性的优点。

本章节 2.1 节的 2.1.1 - 2.1.7 基础理论部分介绍了以 Deepseek 和 Llama 为代表的 transformer 架构的开源大语言本地模型注意力机制学习和运算方式。从本章 2.1.1-2.1.7 的注意力机制本地运算理论基础讲我们可以知道，Transformer 架构的注意力权重机制在无接口读取的条件下会出现严重中间部分信息遗漏问题。本章节 2.1.1-2.1.7 所述的这些 Transformer 类模型的运算注意力算法机制，本课题系统原创的切片处理的长文字处理接口核心算法 Slice-Tree 的算法处理需求提供了理论基础，从而通过本课题的切片处理的长文字处理接口核心算法 Slice-Tree 在实际应用过程中较为完整地读取长段落的文字信息。本章节 2.1.8 介绍了训练层面的 AI-HPC 训练方法，这种训练层面的优化运行方法也为本课题在使用应用层面原创长文字处理接口核心算法 Slice-Tree 的算法提供了一种处理思路的启发。

本章节 2.2 概略地介绍了 Deepseek 和 Llama 为代表的 transformer 架构的开源大语言本地模型部署的自身运行结构层面具有可用性兼容性的优点，本章节 2.1.1-2.1.7 同时具象化介绍 transformer 模型的结构，从而介绍了以 Deepseek 和 Llama 为代表的 transformer 架构的开源大语言本地模型部署的兼容性、可用性。论证以 Deepseek-R1 为代表的 transformer 的开源大语言模型之间具有兼容性的优点，论证以 Deepseek 和 Llama 为代表的开源大模型可以不一定必需昂贵的专用的高性能的服务器硬件，就可以在 Ollama 这种开源的平台上面进行本地的离线运行的理论基础。

第三章 系统前后端的结构与实现

3.1 系统的需求分析

面向 PDF 文档的图表内容智能理解与问答系统的目标是：为 PDF 文档处理需求客户提供本地化基础的保护隐私能力的 PDF 摘要生成平台。注册用户可以在用户同意以后保存用户摘要生成历史记录并删除临时文件夹里面的临时文件，对非注册游客不保存摘要生成历史记录并删除临时文件夹里面的临时文件。因此，本地化的面向 PDF 文档的图表内容智能理解与问答系统应该满足以下需求：

- 1、可以离线地本地化运行大模型。
- 2、具有 PDF 文件的文字和表格的接口。
- 3、可以在本地运行 PDF 摘要人工智能生成。
- 4、管理员可以管理用户信息。
- 5、登录用户可以查看自己的历史记录。
- 6、用户的注册和自己注销功能、修改自己的密码等。

用户可以在本地部署该项目以保障用户使用状态的用户隐私性。在非登陆模式下本地部署不保存文档摘要生成的历史记录，在登陆模式下本地部署保存文档摘要生成的历史记录，管理员用户可以修改和删除不同账户的账户信息。

以帮助师生查找未标注摘要的长文献为例，通过优质论文的大量论文 PDF 数据，自动生成文章内容摘要，为资金实力较弱的学校或学习专业知识的普通学习者提供提供更高质量的文献摘要的文献查询服务，辅助师生查找长文献，提高学习的效率、提高师生的专业知识水平。用户在本地平台或开放平台提交文献，平台通过 PDF 接口与 Ollama 本地离线部署的 Deepseek-R1 开源模型，可以提供本地服务或开放为 WEB 应用，最终用户便可以使用该应用对文献生成 PDF 内容的文字摘要。

用户完成查询之后系统内部缓存的临时文件应被删除以保护用户的隐私。如果用户注册与登陆，用户可以查看或删除自己的文献摘要查询历史记录。如果用户不登陆，则只能临时查询文件而不会保存用户的历史记录，只有注册的用户同意以后才会保存历史记录。

本系统用户的领域模型是：用户在测试模式可以上传 TXT 文档测试摘要生成功能，生成 PDF 文字的摘要，通过边界判断进行文件错误处理，最后可以由用户查看历史摘要。用户在正式情况下可以上传 PDF 文档进行图表内容转换，生成 PDF 文字的摘要，

随后由边界判断进行文件错误处理，最后可以由用户查看历史摘要。同时用户自己的历史摘要进行用户的注册登陆管理和保护中小企业本地部署的无需登陆的隐私模式等模式切换。用例模型如图 3-1 所示。

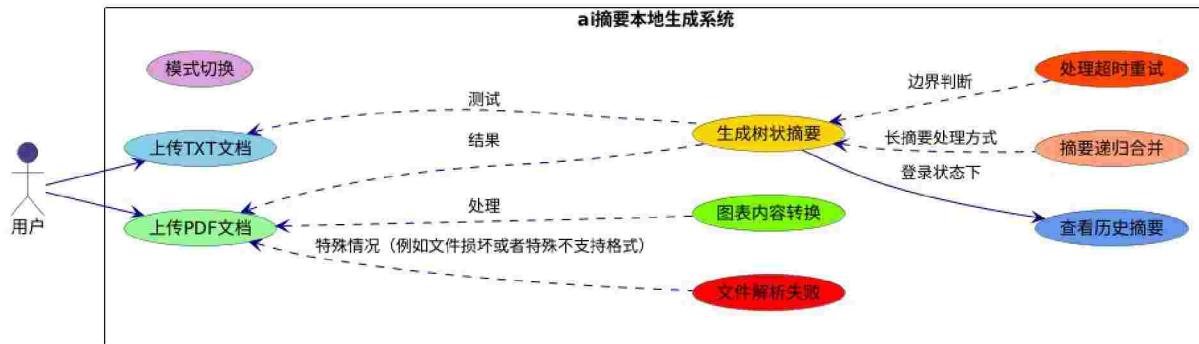


图 3-1 本课题开发的 PDF AI Viewer 运行程序用例模型

本课题是一个 AI 助力数字化转型升级的解决方案，提供了本地离线部署的 PDF 文献较为完整分片阅读的摘要生成能力，通过本地离线存储和处理数据实现基本的临时文档删除隐私保护。

3.2 系统的概要设计

本课题的 Llama 的 PDF 文档处理系统后端主要基于 Vue 2 项目服务集成的 OLLama 技术实现。Vue 2 项目是基于 node.js 开源项目开发而研发的免费、简化、有模板便于开发、开源的系统框架，Vue 2 项目常常用于构建 Java 微服务。开发人员使用 Vue 2 项目以更少的配置构建微服务，Vue 2 项目本身具有成熟模板可以减少开发过程中的 BUG，帮助程序员方便易用地开发软件系统。

3.2.1 系统的模块组成

本系统总计有 3 个本地运行的后端和 1 个本地运行的前端模块。本系统的前端模块是带有模型处理切片树 Slice-Tree 脚本的 Vue 2 前端。

3 个后端模块分别是：

- 1、部署有本地 deepseek 的 OLLama 本地离线服务后端。
- 2、使用本地 MongoDB 开源数据库服务的 node js 后端。
- 3、基于开源模块 npm-PDFreader 改装的 PDF 文件文字和表格内容提取模块的本地 api 实时服务改装的后端。

本系统的模块有 2 种运行模式：

第一种模式是 txt 文档读取的测试模式，使用 Vue 2 前端配备双后端模式，带文本切片树 Slice-Tree 脚本的 Vue 2 前端 + 部署有本地 deepseek 的 OLLama 本地离线服务后

端 + MongoDB 的 java 的 node js 后端。

第二种模式是 PDF 文档读取的正式模式，使用 Vue 2 前端配备三后端模式，带文本切片树 Slice-Tree 脚本的 Vue 2 前端 + Ollama 本地离线服务后端 + MongoDB 的 java 的 node 脚本 js 后端 + 基于公开的开源项目 npm-PDFReader 改编的 PDF 文字图表内容提取后端。本课题开发系统的模块架构如图 3-2。

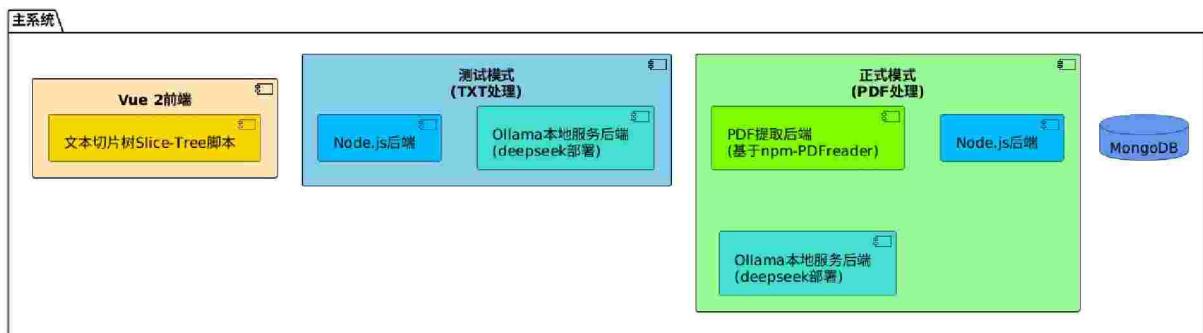


图 3-2 本课题开发的 PDF AI Viewer 程序模块的运行模式模块架构

PDF 文档读取的正式模式处理时，带文本切片树 Slice-Tree 脚本的 Vue 2 前端先通过 npm-PDFReader 改编的 PDF 文字图表内容提取后端，把 PDF 文档的表格和文字提取出来，归一化提取为一串符合格式的规范化文本长字符串 S。

随后，前端内置的文本切片树 Slice-Tree 脚本将字符串 S 切片为长度为 1000 字的子树投入部署有本地 deepseek 的 Ollama 本地离线服务后端生成 100 字摘要，再将每一级子摘要合并投入部署有本地 deepseek 的 Ollama 本地离线服务后端生成下一级 100 字摘要，最后树状合并子摘要生成最终文件内容摘要。

3.2.2 系统的领域模型

本课题开发系统的领域模型如图 3-3，本系统的领域模型是：

Vue 前端具有用户管理模块模式切换接口、文本预处理和摘要展示功能。Vue 前端把文件提交给 PDF 处理器，进行文字提取、图表转换和长字符串生成并返回字符串给 Vue 前端；

PDF 解析流程是：提取文本内容、转换图表描述为文本、生成规范化字符串、可改进的部分是进行异常内容标记。

Vue 前端将长字符串交给 Vue 前端内置的 Slice-Tree 处理器，进行 1000 字长度字符串切片、摘要树合并、递归生成控制；

Slice-Tree 的迭代处理流程是：接收长字符串输入、按照 1000 字切片分组、发起 Ollama 摘要请求、递归合并生成树状摘要。

OLlama 服务进行 Deepseek 模型加载、摘要生成和层级递归的处理；摘要生成规则是：每级输入上限 1000 字、输出上限 100~200 字、保留关键词标记、支持中英文混合。

Vue 前端的文件处理模式有测试模式的 TXT 格式文本读取摘要生成测试功能、正式模式的 PDF 文本读取摘要生成功能。

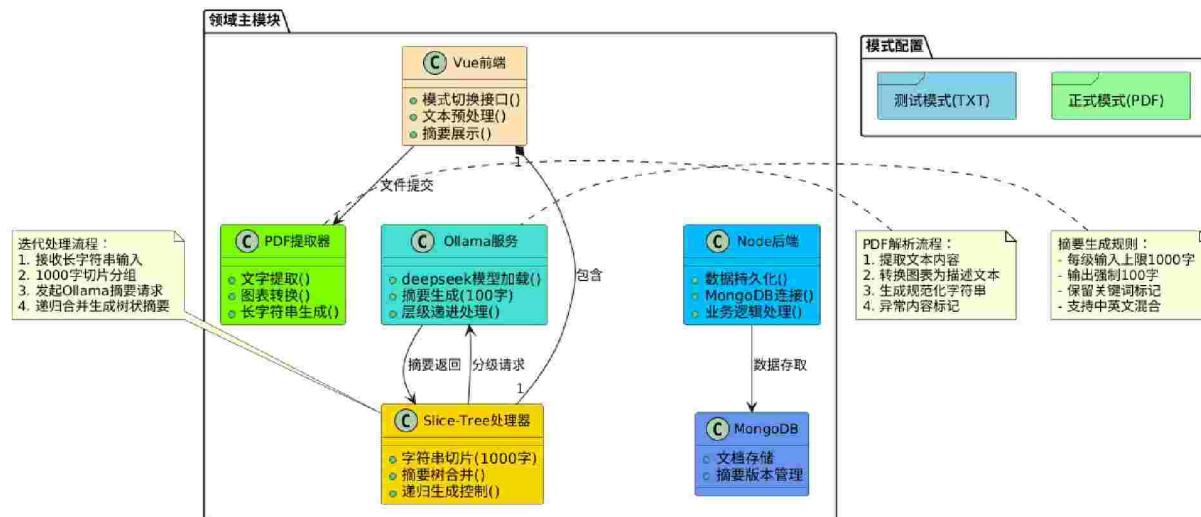


图 3-3 本课题开发的 PDF AI Viewer 运行程序领域模型

无需登陆的游客模式是不留存历史记录的本地离线使用的私密模式，游客模式下无需登录，也不留存游客模式历史摘要记录。用户账户用于存储登陆模式下产生的摘要生成任务历史记录，用户可以查看或者删除自己的历史记录。管理员 admin 是初始设定的，管理员可以删除或者修改任一账户的密码、删除其历史记录(也可以修改自己的密码)；其他用户只能修改自己的密码，如果忘记密码需要联系管理员修改密码。

3.2.3 系统的操作契约

本课题开发的 PDF AI Viewer 的操作契约分为三个部分：主模块操作契约、用户的权限与等级操作契约、文档处理模式操作契约三部分。

1、主模块操作契约：

用户管理契约类：进行身份验证；管理管理员与普通用户的权限差异；进行密码修改和注册注销。本系统在用户管理契约的细化设计和改进方向是：密码哈希加密传输和存储、会话令牌自动刷新、重要操作二次验证、密码保护等机制。

文件处理契约类：文件处理前置条件。文件处理后置保证。文件处理契约的通用文

档处理规则是：文本编码自动检测、临时文件隔离存储、临时文件隔离存储、处理进度实时反馈等。

2、不同的用户权限等级是：

游客模式用户管理契约：约束条件有不存储历史记录、无需登录、保护用户安全隐私。

普通用户模式用户管理契约：权限包括查看自有记录、删除个人记录或删除账户、修改登陆密码。

管理员模式用户管理契约：管理员权限包括：重置任意密码、删除所有记录或任一账户、查看操作日志。

3、文档处理模式使用的操作契约是：

测试模式处理 TXT 文件的文件处理契约：文件处理前置条件是：文件类型.TXT，文件大小 $\leq 10MB$ 。文件处理后置保证是：生成任务 ID，触发字符串智能摘要生成的文字和表格内容的自动问答切片树 Slice-Tree。

正式模式处理 PDF 文件的文件处理契约：文件处理前置条件是：文件类型.PDF，包含文本层，GPU 内存 $\geq 500MB$ 。文件处理后置保证是：图表转描述文本，生成规范字符串，触发字符串智能摘要生成的文字和表格内容的自动问答切片树 Slice-Tree。

本系统的操作契约的内容如图 3-4 所示：

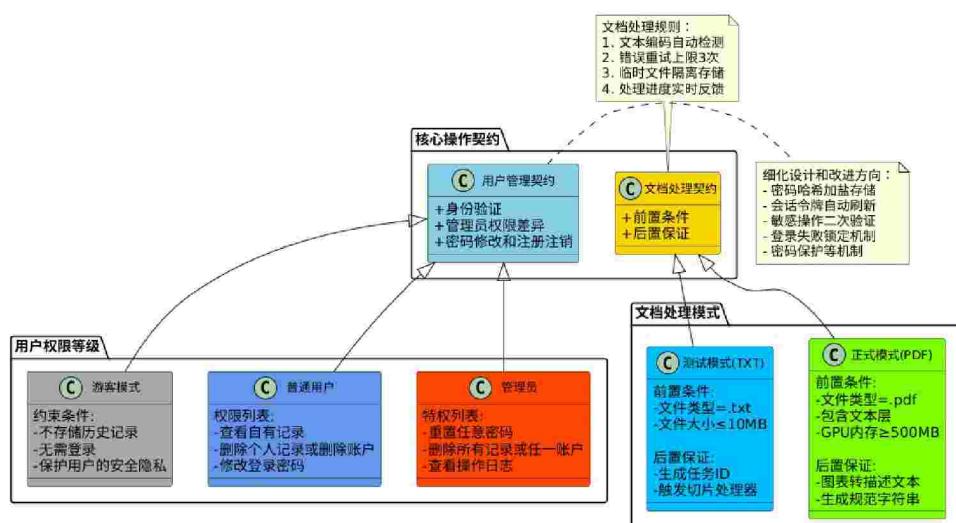


图 3-4 本课题开发的 PDF AI Viewer 运行系统操作契约

3.2.4 系统的时序设计

本课题开发的系统各个模式时序设计如下：

本系统综合分为测试模式、正常模式、用户管理状态和保护中小企业隐私信息的游客模式执行，这几个主要模式有各自的系统运行时序，这些功能在各自模式下的系统执行时序在时序设计里面一并展示。

测试模式的用户时序是：

用户先对 Vue 前端上传 TXT 文件，随后在 Vue 前端查看加载文本。Vue 前端生成字符串 S。Vue 前端随后执行切片树 Slice-Tree 脚本形成切片。

Vue 前端进入递归执行的切片树 Slice-Tree：Vue 前端发送 1000 字文本切片给 OLLama 开源本地服务，OLLama 开源本地服务生成 100~200 字中英文摘要，Vue 前端合并子摘要。

用户在登陆情况下，Vue 前端提交最终摘要给 Node.js 后端由 MongoDB 数据库保存。

正式模式的用户时序是：

用户先对 Vue 前端上传 PDF 文件，随后在 Vue 前端对 PDF 加载后端请求解析。PDF 加载后端返回长字符串 S。

Vue 前端随后执行切片树 Slice-Tree 脚本形成切片。Vue 前端进入递归执行的切片树 Slice-Tree：Vue 前端发送 1000 字文本切片给 OLLama 开源本地服务，OLLama 开源本地服务生成 100~200 字中英文摘要，Vue 前端合并子摘要。

用户在登陆情况下 Vue 前端提交最终摘要给 Node.js 后端由 MongoDB 数据库保存。

用户管理模式的用户时序是：

用户对 Vue 前端发起登陆、注册请求。

Vue 前端通过连接 Node.js 后端验证用户身份。

如果是无管理权限常规用户，Node.js 后端根据用户身份对 Vue 前端返回用户令牌；Vue 前端用户可以通过 Node.js 后端进行历史记录操作；Node.js 后端对 MongoDB 数据库执行查询或删除操作。

如果有管理权限的管理员用户，Node.js 后端根据用户身份对 Vue 前端返回管理员令牌；Vue 前端用户可以通过 Node.js 后端进行密码重置和记录删除操作；Node.js 后

端对 MongoDB 数据库执行管理员权限的操作。

未登陆状态本地运行游客模式不存储历史记录，以保护中小企业隐私，游客模式的本地处理用户时序是：

游客通过 Vue 前端未登陆状态执行本地的隐私离线模式。

Vue 前端随后执行测试模式或者正式模式的 TXT 文件或者 PDF 文件本地摘要生成处理工作。

游客的用户通过 Vue 前端查看不保留历史记录的处理结果。

Vue 前端到 PDF 提取后端的 PDF 文件处理规范是：图表内容转为描述文本。自动编码统一处理。

Vue 前端与 OLLama 开源本地服务连接执行的摘要生成规则是：输入 ≤ 1000 字的中英文字符。输入强制 100 字精简。保留关键词标记。支持中英文混合。

综上所述，本系统开发的系统各个模块在各个模式下运行的时序设计图如图 3-5 所示：

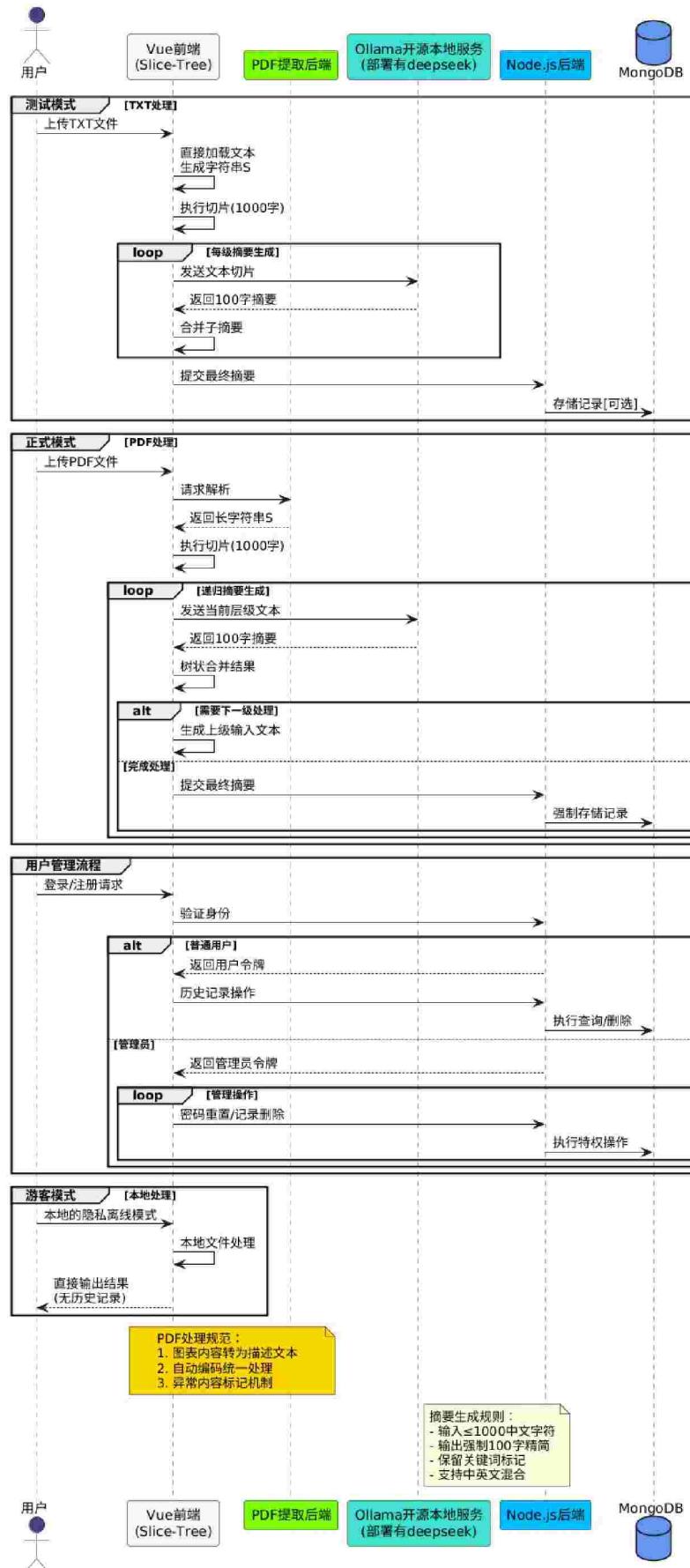


图 3-5 本课题开发的 PDF AI Viewer 运行系统时序图

3.2.5 系统的开发工具

OLlama 是一个便于开源大模型在本地部署的开源部署工具，这种自身开源的开源平台在起初用于部署 Llama 开源模型，也可以用于部署 DeepSeek-R1-1.5B 本地模型。OLlama 在本地部署，适合用户编辑开发和保护用户隐私，它的本地化服务配置具有很高的可塑性。OLlama 作为高级语言开发的开源工具还有多种平台支持，它可以在 windows、ubuntu 和 mac 三种操作系统上面运行，具有良好的可扩展性。OLlama 还支持和 Vue 2 项目上面的接口配合使用。OLlama 还具有多功能，支持多种大模型的使用。在 Vue 2 项目里面 javascript 脚本进行本地服务器通信，就可以使用 OLLama 的模型本地化运行对接 vue 2-node.js 模块的 api 接口了。

Vue 2 的项目通常可以使用 Visual Studio 进行编辑和配置。使用 Visual Studio 的 IDE 编辑配置 Marven 和 Vue 2 项目。Vue 2 项目的优势有：基础地启动 Spring 项目；含有 Node.js -axios 开源路由器这些环境；提供 starter 模板便于设置 Node.js 开源 api 本地路由运行平台；内置模板可以对 Spring 容器默认配置；还会支持 java 配置；Vue 2 项目还具有 debug 功能可以监视运行时的项目状态。

本程序系统使用 HTTP 协议在前后端之间进行关联和通信，这样可以简化应用程序编写方式，减少应用程序设计的漏洞。具有模块化组合的特征，具有高内聚和低耦合的应用程序编写优势。这种微服务部署比较灵活，既可以在线布置也可以离线部署，本系统是一种离线部署的微服务程序。

3.3 系统的数据结构

1.mongo-db 后端的数据库概要是：

(1) 用户功能 (User)

- 功能是管理用户账户信息及权限。
- 属性是：
 - user_id (用户 ID, 主键)
 - username (用户名, 唯一)
 - password (加密存储的密码)
 - loginstatus (登录状态)
 - adrole (权限: 普通用户/管理员)
 - recordlist (用户记录)

(2) PDF 文档 (PDFDocument)

- 功能是存储用户上传的 PDF 文件及解析状态。
- 属性是：
 - document_id (文档 ID, 主键, 上传时间)
 - user_id (外键, 关联用户)
 - file_path (文件存储路径)
 - upload_time (上传时间)
 - parsed_status (解析状态: 未解析/解析中/解析完成/解析失败)
 - temporary_flag (是否为临时文件: 是/否)
 - metadata (文档元数据, 如页数、大小、创建时间等)

(3) 系统配置 (SystemConfig)

- 功能是管理模型接口、隐私策略等配置。
- 属性是：
 - config_id (配置 ID, 主键)
 - model_type (大模型类型, 如 Llama-7B)
 - model_params (模型运行参数, JSON 格式)
 - privacy_policy (隐私策略, 如临时文件保留时长)
 - api_key (Ollama 接口密钥, 加密存储)

实体关系：

- 用户与 PDF 文档为一对多关系（一个用户可上传多个 PDF 文档）。
- PDF 文档与图表内容为一对多关系（一个文档包含多个图表）。
- 用户与问答记录为一对多关系（用户可生成多条记录，非注册用户记录临时存储）。
- 图表内容与问答记录为一对多关系（一个图表可关联多个问答记录）。

2. 数据样例 MongoDB 文档

(1) 用户文档 (User Collection)

```
{
  "user_id": "U001",
  "username": "user1",
  "password": "alb2c3d4...",
  "loginstatus": true,
  "adrole": "false"
}
```

(2) PDF 文档的存储 (PDFDocument Collection)

```
{
  "document_id": "D001",
  "user_id": "U001",
  "file_path": "/storage/documents/D001.PDF",
  "upload_time": "2025-05-20T14:35:00Z",
  "parsed_status": "解析完成",
  "temporary_flag": false,
  "metadata": {
    "pages": 10,
    "size": "2.5MB",
    "created_time": "2025-05-18T09:00:00Z"
  }
}
```

(3) 索引:

- 用户表: user_id (主键索引)、username (唯一索引)。
- PDF 文档表: document_id (主键索引)、user_id (外键索引)。

(4) 隐私的保护功能:

- 临时文件处理: 可以非注册用户的 PDF 文档和问答记录标记为 is_temporary: true, 系统定期清理 (如 24 小时后自动删除)。
- 数据加密: 用户密码、API 密钥等敏感信息使用 AES 加密存储。
- 访问控制: 通过用户角色 (role 字段) 限制管理接口权限。

(5) 数据流:

- 用户 (User) 上传 PDF 文档 (PDFDocument)
- PDF 文档 (PDFDocument) 包含图表内容 (Chart)
- 用户 (User) 生成问答记录 (QAHistory)

- 图表内容 (Chart) 关联问答记录 (QAHistory)

数据原型通过 MongoDB 文档结构实现灵活存储，从而支持高并发读写和程序服务功能动态扩展。通过索引优化和隐私保护策略，从而可以保障本系统高效、安全地运行，满足用户对 PDF 图表内容智能理解与问答的需求。

3.4 系统的后端结构与实现

本节的 2 个子节介绍系统的 2 个后端：本地 MongoDB 数据库访问后端和 Ollama 人工智能模型开源平台后端的实现。功能更加复杂和重要的 PDF 文字和表格读取后端的结构与实现，见后文第 4.2 节部分对 PDF 文字和表格读取后端的功能、设计和实现的详细的说明。

3.4.1 本地 MongoDB 数据库访问后端

数据库访问后端核心文件是 server.js，导入 mongoose 的 MongoDB 支持模块，设置 api 自身端口号为 8080，这个 api 专用对接目标为 8008 本地前端端口，连接数据库操作脚本 newuseruse.routes.js。数据库操作脚本 newuser.controller.js 引入控制器脚本 newuser.controller.js。控制器脚本 newuser.controller.js 导入本数据库模型脚本 /models/index.js，MongoDB 的本地数据库模型脚本为 newuser.model.js。

本地 MongoDB 数据库访问后端的模块架构设计是：

server.js 自身的 api 端口是 8080，是后端核心，与监听前端端口 8008 对接。

newuseruse.routes.js 是路由模块，定义 api 端点，处理本地服务 http 请求。

newuser.controller.js 是控制器模块，进行业务逻辑处理和数据验证工作。

/models/index.js 是模型入口模块，导出数据库连接模型 newuser.model.js。

数据库模型 newuser.model.js 是用户的数据库模型，进行用户的数据库 Schema 定义，管理 MongoDB 数据库存储的数据结构。

该后端的模块架构配置如图 3-6 所示：

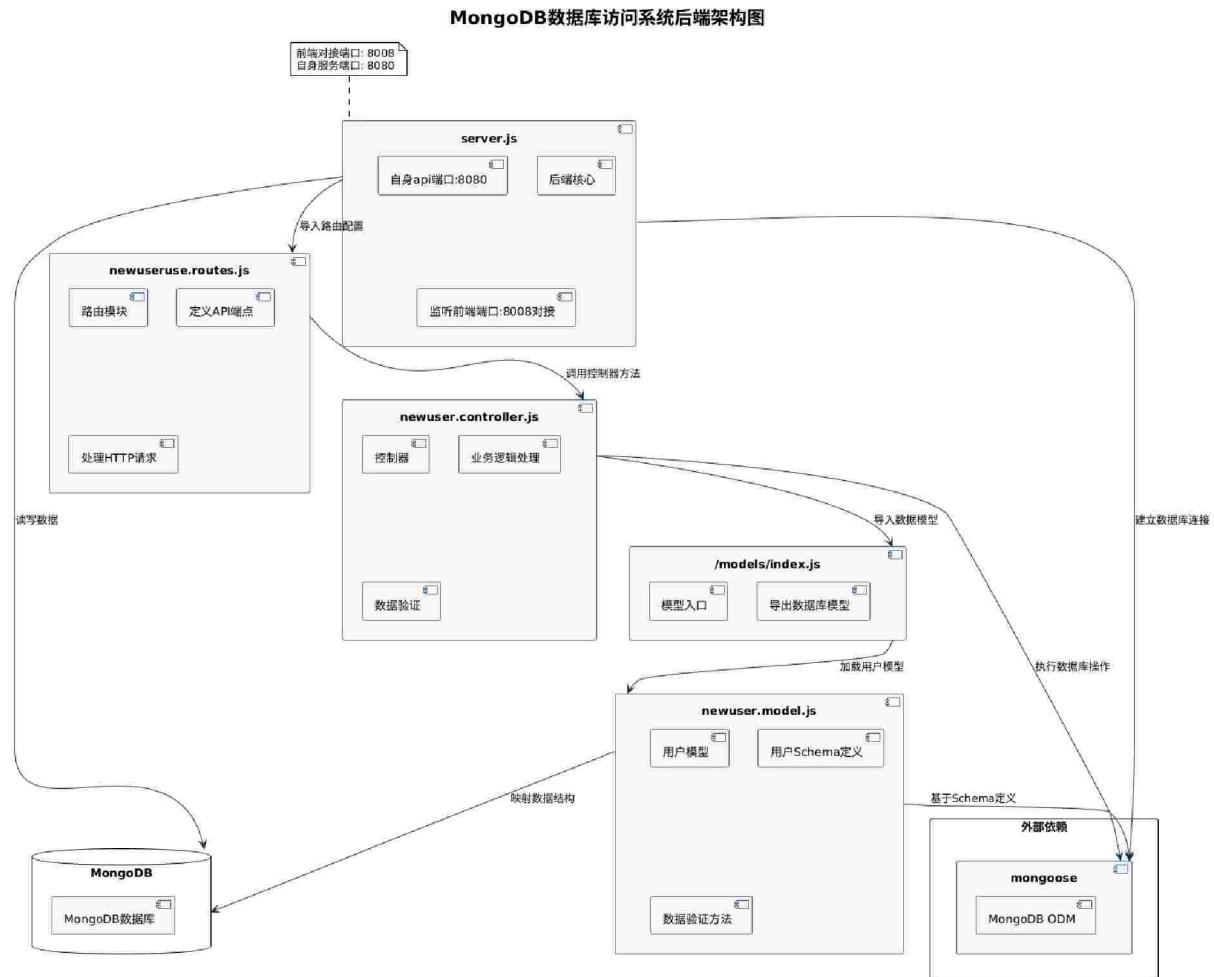


图 3-6 本课题开发的 PDF AI Viewer 数据库访问后端的模块架构图

数据库访问后端的领域模型是：

系统主模块类 `Server`。`api` 参数设定为 `port: 8080; frontendPort: 8008`。属性函数设定为：启动服务器 `startServer()`；连接数据库 `connectToDatabase()`；创建管理员 `createAdmin()`。

系统路由器类 `NewUserRoutes`。属性函数设定为：存储新用户 `post("/", newusers.create)`；查找全部用户 `get("/", newusers.findAll)`；根据令牌查找用户 `get("/:id", newusers.findOne)`；根据用户名查找用户 `get("/username/:username", newusers.findByUsername)`；根据令牌更新用户 `put("/:id", newusers.update)`；根据令牌删除用户 `delete("/:id", newusers.delete)`。

数据库控制器类 `NewUserController`。属性函数设定为：创建新用户 `create(req, res)`；查找全部用户 `findAll(req, res)`；查找一个用户 `findOne(req, res)`；按照用户名查找用户 `findByUsername(req, res)`；更新存储用户 `update(req, res)`；按照令牌更新用户

`findByIdAndUpdate(id, req);` 删除用户 `delete(req,res)`。

`ModelIndex` 类：引出数据库模型 `NewUserModel` 类。

数据库模型 `NewUserModel` 类：`userSchema` 数据库数据结构属性；`FileHistory` 数据库数据结构属性。

数据库接口模块 `MongooseModule` 类：`Schema` 数据库操作结构属性；`Model` 数据库模型结构属性；`connect()` 数据库连接方式属性。

数据库MongoDB访问后端领域模型

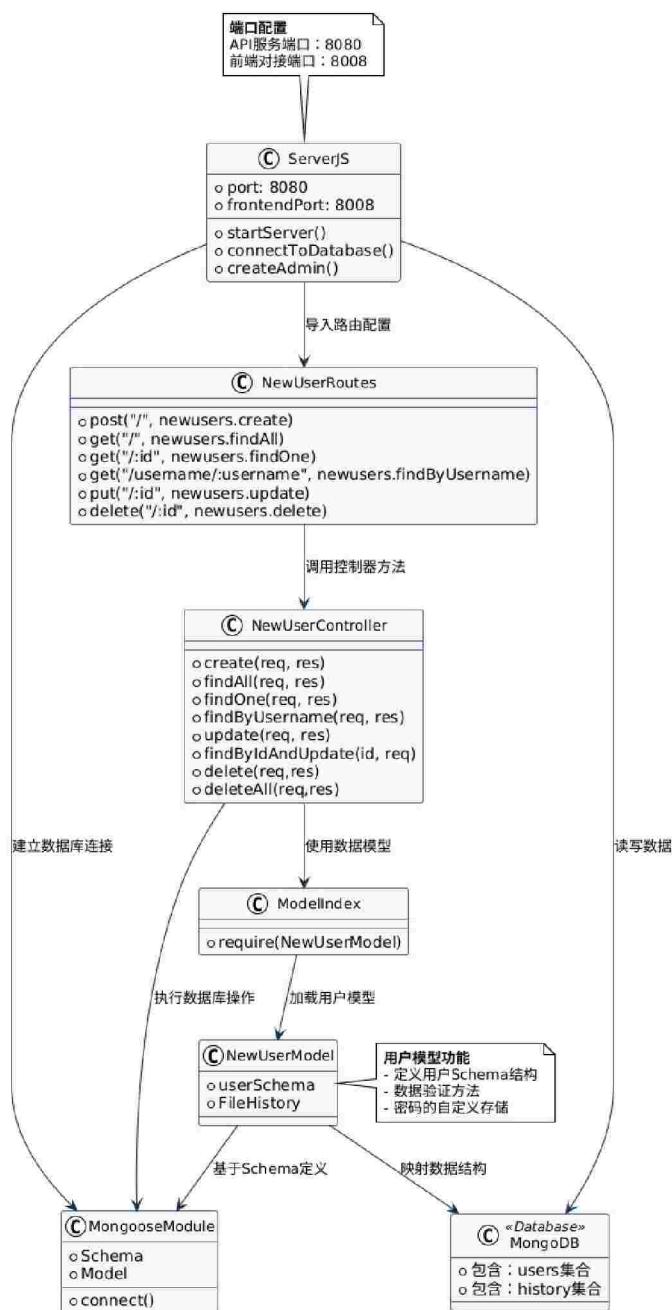


图 3-7 本课题开发的 PDF AI Viewer 数据库访问后端的领域模型

数据库后端的时序设计是：

数据库后端主要功能分为 3 个部分：用户的注册、用户的登录和用户的历史记录查询。

数据库后端执行主要功能之前后端的启动时序：

- 1、建立自身 8080 服务端口。
- 2、通过 bat 脚本启动 MongoDB 数据库服务，并与 MongoDB 数据库服务建立连接。
- 3、通过自身端口号为 8080 的 api 与端口号我 8008 的前端自动连接。

数据库后端执行用户的注册和登录与历史记录查询的功能时序：

- 1、server 接受到 8008 端口发送注册请求。
- 2、server 把服务请求转到 NewUserRoute。
- 3、NewUserRoute 向 NewUserController 发送 create 请求。
- 4、NewUserController 向 NewUserModel 发送创建用户请求。
- 5、NewUserModel 向 Mongoose 发送 Schema 数据库请求。
- 6、Mongoose 向 MongoDB 查询数据。
- 7、MongoDB 向 Mongoose 返回数据。
- 8、Mongoose 向 NewUserModel 返回用户数据。
- 9、NewUserModel 向 NewUserController 返回用户对象。
- 10、NewUserController 向 NewUserRoute 返回用户对象。
- 11、NewUserRoute 向 Server 返回用户对象。
- 12、Server 通过 api 向 Vue 前端返回用户数据。

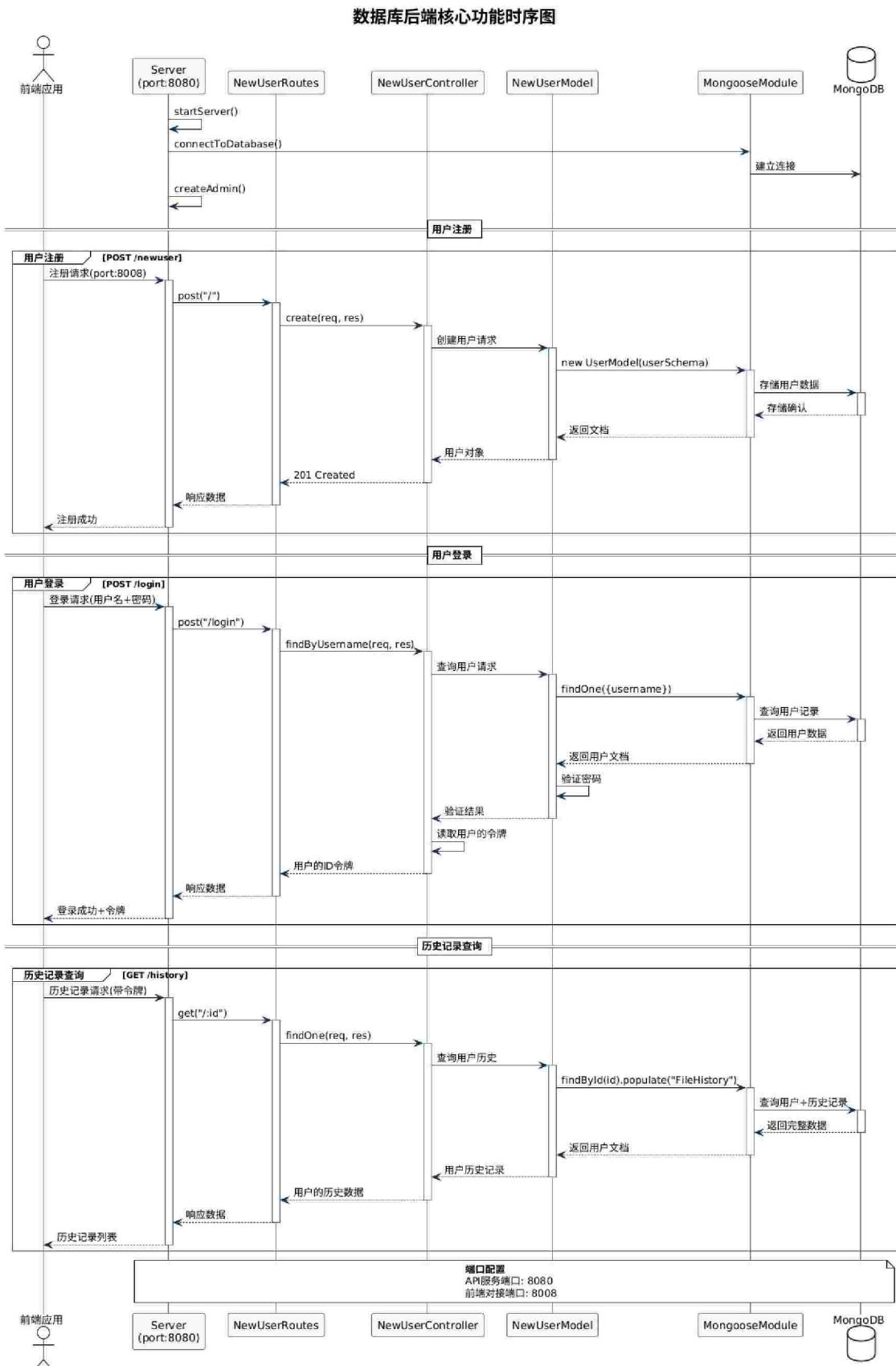


图 3-8 本课题开发的 PDF AI Viewer 数据库访问后端的功能时序图

例如数据库后端控制器脚本 newuser.controller.js 进行查询工作例如 findByUsername 函数通过用户名查找用户信息，用于注册或登陆操作和管理员用户信息查询操作：

```
exports.findByUsername = async (req, res) => {
  try {
    const user = await PDFFile.findOne({ username: req.params.username });
    if (!user) {
      return res.status(404).json({
        error: `User ${req.params.username} not found`
      });
    }
    res.json(user);
  } catch (err) {
    res.status(500).json({ error: err.message });
  }
};
```

3.4.2 本地的 OLLama 人工智能模型开源平台后端

系统的 OLLama 开源本地运行平台后端是通过本地 bat 脚本启动的，本地开源平台 OLLama 的 bat 脚本的启动初始设定为以 400MB 小内存和 CPU 模式运行，设置本地端口号为 8011，设置最大响应等待时间 15 秒，设置窗口大小为 4096 字节。

```
set OLLAMA_MAX_VRAM=402653684
set OLLAMA_MODELS=D:\oLlama\MODELS
set OLLAMA_HOST=http://localhost:8011/
set OLLAMA_NUM_PARALLEL=5
set OLLAMA_GPU_UTILIZATION_TIMEOUT=15
set OLLAMA_NUM_GPU=0
set OLLAMA_MAX_CTX=4096
set OLLAMA_ORIGINS=*
oLlama serve
```

OLLama 本地后端输入的 prompt 字符串使用提示语句问答的方式输入对文件切片生成总结的字节流指令，问答的指令是：

[prompt: 请仔细阅读用户在"用户档开始"和"用户档结束"之间的绝大多数文字提取关键信息并生成一个 200 字以内的在"用户档开始"和"用户档结束"之间的绝大多数文本的简体中文总结][prompt: 错误示例: Hello, World, or English think weight higher → 正确做法: 生成一个 200 字以内的简体中文总结。保留英语词汇: CNN, NASA, GPT-4, TaPERA, ApachePDF 等短术语单词并将保留英语词汇写在括号 () 内][用户档开始]/*用户文件输入的切片内容*/[用户档结束][prompt: 请仔细阅读用户在"用户档开始"和"用户档结束"之间的绝大多数文字文本生成一个 200 字以内的在"用户档开始"和"用户档结束"之间的绝大多数文本的简体中文总结]

本地 OLLama 开源运行平台的服务后端的 api 设定为 8011，它通过 java 脚本的 JSON.stringify 连接方式访问，指定本地化运行模型是: deepseek-r1:1.5b。其中本地开源 api 的输入字符串变量为: this.userInput，输出为字符流 const response:

```
async handleSubmit() {
  try {
    //此处省略大段的初始化代码
    const response = await fetch('http://localhost:8011/api/chat', {
      method: 'POST',
      headers: { 'Content-Type': 'application/json' },
      body: JSON.stringify({
        model: 'deepseek-r1:1.5b',
        messages: [{ role: 'user', content: this.userInput }],
        stream: true
      })
    });
  }
}
```

3.5 系统的前端结构与实现

登陆界面前端主文件为 App.vue，提供菜单和登陆的服务，通过索引 router.js 链接到如下几个 vue 模块：使用流程模块 Help.vue，文件处理模块 Process.vue，历史记录列表 PDFFilelist.vue，历史记录查看 PDFFile.vue，用户登陆模块 Login.vue，用户注册模块 Register.vue，用户信息修改模块 User-change.vue，用户信息列表选择模块 Edit-user.vue。

3.5.1 系统的前端概要设计

系统的前端模块架构设计是：

App.vue 是主应用入口，负责通过路由导航控制提供菜单服务、负责用户的 ID 令牌数据在不同子模块之间进行令牌传递服务。

索引 router.js 是路由配置中心，进行路径映射。

Help.vue 是系统的操作帮助页面，提供系统配置信息和操作指南。

Process.vue 是 PDF 文件处理页面，负责 PDF 文件处理服务。

PDFFilelist.vue 是历史记录列表，负责展示历史记录文件概要的列表。

Login.vue 是用户登录模块，负责身份验证。

Edit-user.vue 是用户信息管理模块，负责选择用户信息进行编辑。

Register.vue 是注册模块，负责用户信息创建。

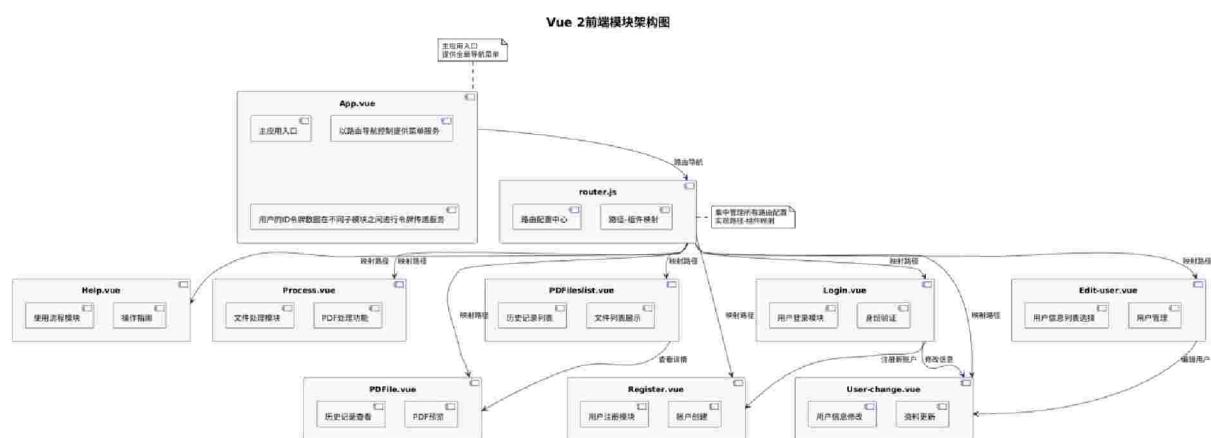


图 3-9 本课题开发的 PDF AI Viewer 前端的模块架构图

本系统前端的领域模型是：

主应用 App 类：主应用入口属性、提供菜单服务属性、管理路由导航属性、传递用户令牌属性。

路由 RouterJS 类：路由配置中心属性、组件的路径映射属性、参数传递属性。

帮助信息类 HelpVue：操作帮助页面属性、系统配置信息属性、操作指南文档属性。

文件处理类 ProcessVue：PDF 文件处理属性、文件上传/转换属性和文件智能摘要生成接口 Slice-Tree、处理结果展示属性。

历史记录展示类 PDFFilelistVue：历史记录列表属性、文件概要展示属性、文件选择功能属性。

登录界面类 LoginVue：用户登录模块类属性、身份验证属性、令牌获取属性。

用户信息编辑类 EditUserVue：用户信息管理属性、用户选择属性、信息编辑属性、权限控制属性。

注册类 RegisterVue：用户的注册模块属性、账户创建属性、用户名验证属性。

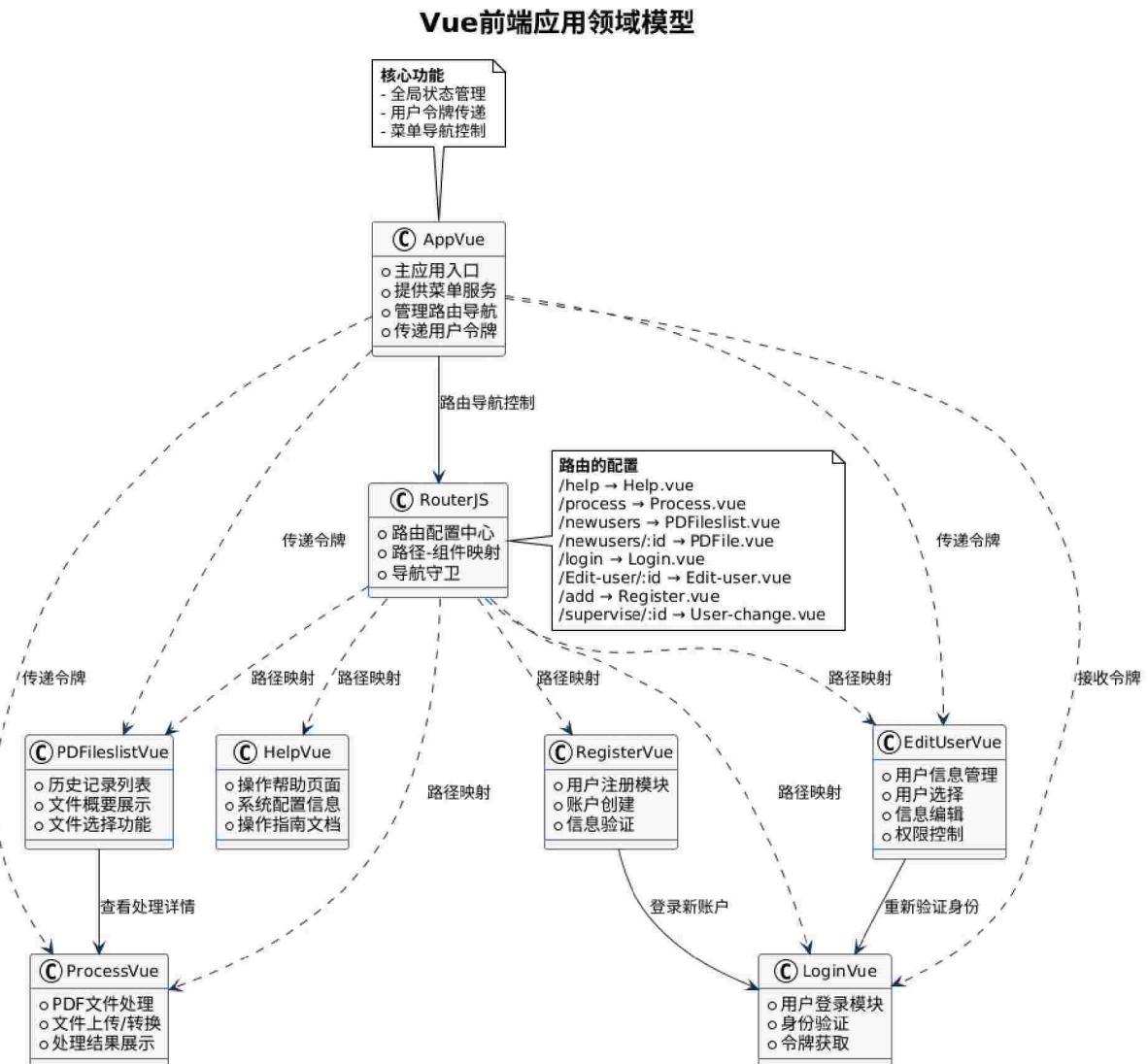


图 3-10 本课题开发的 PDF AI Viewer 前端的模块领域模型

3.5.2 系统的前端配置与用户管理功能实现

在 vue.config.js 设置前端本地服务地址为 8008。

```

module.exports = {
  devServer: {
    host: '0.0.0.0',
    port: 8008,
    open: true,
    hot: true,
  }
}

```

主项目文件 App.vue 同时导入数据库后端 api 脚本 PDFFileDataService.js 与数据库后端连接。切片树 Slice-Tree 处理脚本位于文件处理模块 Process.vue。

前端进行用户管理功能的实现：

用户注册和修改用户名需要查找是否存在用户名并且驳回重复用户名：

```
<label for="username">User name</label>
  <input
    type="text"
    class="form-control"
    id="username"
    required
    v-model="newuser.username"
    @input="checkUsernameAvailability"
  />
  <div v-if="isChecking">checking...</div>
  <div v-if="usernameError"
style="color:red">{{ usernameError }}</div>
  <div v-if="isUsernameAvailable"
style="color:green">User-name available</div>
```

登陆用户在 Login.vue 文件通过 Logintofather 脚本函数与主模块 App.vue 传递登陆用户的数据库 ID。登陆用户在 Login.vue 通过不间断接收 watch 管道”Fatherloginwindow”决定是否显示出登陆状态窗口。

```
props:["Fatherloginwindow"],
watch:{
  Fatherloginwindow(sv){
    this.Fatherloginstatus = sv; }
},
Logintofather(val) {//send message to father
  let loginson = {
    sonloginaccount: val
  };
  this.$emit('toFatherLogin',loginson);//Send data to
father. After select case automatically emit showCityName case
},
```

用户在 Edit-User.vue 进行用户信息的删除和修改工作，该文件内含有脚本进行管理员权限判断，管理员的权限可以删除和修改全部用户信息，用户只能删除或修改自己的信息。Edit-User.vue 内的用户权限判断：

```

Usershow() {
    if(this.$route.params.id != '0'){
        if(this.currentLoginUser.adrole == true){
            PDFfileDataService.getAll()
            .then(response => {
                this.newusers = response.data;
                console.log(response.data);
            })
            .catch(e => {
                console.log(e);
            });
        }
    }
    else{
        this.newusers=[this.currentLoginUser];
    }
}
else{
    return 0;
}
},

```

3.5.3 系统的前端文件处理功能实现

第四章所述的 PDF 或 TXT 文件上传与处理和切片树核心处理脚本 Slice-Tree 功能模块位于 Process.vue 前端模块的内部。对前端长文本的处理方式切片树 Slice-Tree 的原理原则、和算法实现，见本文后续章节第 4.1 节对切片树 Slice-Tree 的切片接口算法的功能、设计和实现的详细的说明。

3.6 本章的内容小结

本章节使用软件工程的方法进行组织，介绍了本课题的程序设计的结构和功能、以及对应结构和功能的实现方式。

本章的 3.1 节系统的需求分析，该节介绍了本课题系统的功能需求和用例模型等。

本章的 3.2 节概要设计用软件工程的方法介绍了本课题设计系统使用的软件架构概要说明：本章第 3.2 节的 3.2.1 节介绍了系统的模块组成架构和系统各个模块各自承担的功能；本章的 3.2.2 节介绍了系统的领域模型，包括了系统的各个种类功能，在各自模块实现各自功能的范围；本章的 3.2.3 节介绍了系统实现各项功能的操作契约与处理模式；本章的 3.2.4 节介绍了系统实现各种功能、进行程序操作的程序时序设计；本章

的 3.2.5 节介绍了系统的软件开发工具特点。

本章的 3.3 节介绍了系统使用 MongoDB 进行历史记录存储和用户管理的数据结构和样例。

本章的 3.4 节的 3.4.1 和 3.4.2 分别介绍了系统 2 种后端的程序设计结构和后端功能的结构设计与实现方式。系统的第 3 个后端也就是 PDF 文件读取后端的功能和结构设计方式见论文后文第四章第 4.2 节的具体说明。

本章的 3.5 节介绍了系统的前端程序设计结构和前端的功能实现方式。对系统前端的文件处理核心算法切片树 Slice-Tree 接口的设计和实现，参见下一章第 4.1 节对切片树 Slice-Tree 的切片接口算法的功能、设计和实现的详细的说明。

第四章 系统的模型处理接口 Slice-Tree 算法与 PDF 读取接口

本章节将会介绍系统的前后端核心处理算法需求、前后端核心算法的目标原理和前后端核心算法的算法运行方法。本章节第 4.1 节部分将会分析系统前端核心的文件处理接口算法 Slice-Tree 的算法需求、算法的目标原则、算法的算法方法结构。本章节第 4.2 节部分将会分析系统后端核心的 PDF 文件的文字和表格读取的算法需求、算法的目标原则、算法的算法方法结构。

4.1 本地小体量模型长文字处理切片树接口 Slice-Tree

本论文前文的第二章从理论基础方面分析了机器学习大模型计算在各层存在不均衡性，这种不均衡读取是提高机器学习概率判断准确性的必然要求，但是这种不均衡性同时必然造成了大模型在无接口条件下直接读取长文本必然存在较多的信息遗漏问题。

4.1.1 构造切片树处理接口的需求和原因

由于机器学习的语言模型的注意力运算机制结构理论和运算方法必然为了逻辑判断准确性，读取运算各级都采用内在不均衡特征，本地部署的模型需要使用较长文字的读取和处理接口结构，例如本课题系统原创的切片处理的细化长文字处理接口算法 Slice-Tree，才可以应用中较为完整地读取长段落的文字信息、减少机器学习读取长文本的信息遗漏问题。

直接使用大模型输入复杂信息时容易产生幻觉造成答案错误，尤其是当任务涉及推理时经常会产生幻觉造成严重的处理错误。当 PDF 文件过长时，transformer 变换器模型就会出现阅读范围过小的问题。ScaleFS 大文件服务器级操作系统使用的服务器文件分级树状处理方式、TaPERA 接口和 AI-HPC 这种分解式训练方法处理思路为本课题的系统提供了一个处理较长 PDF 文件的启发。

由于大模型直接读取字符串长度受到模型本身尺寸的限制和运算性能的限制，因此可以使用减少单次运算长度、延长处理时间的方式在性能相对低于服务器的常规计算机上面运行 Deepseek-R1:1.5B 小体量模型进行相对更加廉价容易部署的长文本处理，本课题的研究不只是适用于 PDF 文件，而且还为相对廉价机器进行本地化的长文本处理提供了一种可行的解决思路。

4.1.2 切片树处理方法的目标和原则

本课题使用切片树的设计目标是：通过减少同一个本地小规模模型单次输入的字符串长度规模 Len，减少本地小规模模型单次读取字符串产生的信息遗漏 FN。

如果切片长度过短，容易导致本地模型难以理解切片的上下文情况，从而造成模型输出偶然性过大、降低模型输出的准确性；如果切片长度过长，会导致模型算法自身逻辑推理需要产生的不均衡性加上模型的规模限制，导致模型处理过长文本时出现信息遗漏。因此，应该选取合适的模型切片长度。

需要注意的是，根据理论判断和本课题实施的实验分析，切片树的最佳切片大小并不等于模型在本地本机环境下的最大允许的单位词数量 token 长度 2048：真正的效果均衡性最好的切片字符数是一个比模型最大允许 token 词汇长度 2048 小得多的字长度 1000 的字符串。根据作者在第 5.2 章节进行的切片树切片大小的性能评估实验，切片效果最佳的长度为 1000 字 char 长度，此时 token 词汇数量在 500-800 之间。因此，理论和实验确定的性能最佳的 token 词汇处理状态为 500-800 token 词汇即 1000 字长度的字符串，实验和理论确定的最佳切片长度远小于模型在本地环境下允许运行读入的最大 token 词汇支持数 2048。

本课题切片树的基本切片设计原则是：包括提示词和长问答提示语句 prompt 在内，单次输入本地机器学习模型的文字总规模小于机器学习本地模型单次可以以最少遗漏读取的最大读取窗口规模的一半，通过适当地缩短机器学习本地模型的输入字符串长度，提高模型对字符串输入的信息读取的完整性。

4.1.3 切片树的构造和生成方法

本课题使用切片树状总结方式生成摘要。

本课题采用的 PDF 后端处理方法方式是：将 npm-PDFreader 读取的 PDF 文件文字和表格段落内容将较长 PDF 文件首先直接通过 npm-PDFreader 开源程序后端服务，提取出文字内容。

模型的切片长度不宜过长也不宜过短，过短的切片容易导致本地模型难以理解切片的上下文而造成模型输出偶然性过大和准确性下降；如果切片长度过长，会导致模型算法自身为了保障逻辑能力的算法读取不均衡性加上模型的规模限制，导致模型处理过长文本时出现信息遗漏。因此，模型切片长度选取是重要的步骤。模型的切片长度选取参见第 5 章的 5.2 节模型的本地运行实验测试通过模型信息遗漏率的程序测试和稳定性测试，选取合适的切片长度。

根据第 5 章的 5.2 节模型的本地运行模型信息遗漏率的程序测试和稳定性实验运行测试，选取合适的切片长度为 1000 字切片可以兼顾模型的低信息遗漏率和较高的本地运行输出稳定性。

根据 4.1.2 介绍的切片树的基本切片设计原则是：包括提示词和长问答提示语句 prompt 在内，单次输入本地机器学习模型的文字总规模小于机器学习本地模型单次可以以最少遗漏读取的最大读取窗口规模的一半。本课题系统在本机低配置运行的读取窗口为 2048 个 token 词汇，因此不超过读取窗口一半长度即 1000 个 token 词汇长度，1000 个 token 词汇长度的中文在 1500 字左右，本系统切片树的窗口大小取 1000 字中文字符长度作为输入切片窗口的大小判定实例。此处的 1000 字切片长度不是 1000 个 token 词汇，而是 1000 个中英文字符，合 500-800 个 token 词汇。

本课题切片方法是将段落进行 1000 字为单位的自顶向下的切片读取，摘要生成使用自底向上的树状生成。根据字符串长度 y 确定切片树级数为：

$y \geq 1000$ 且 $y < 300000$ 时：

$$x = 1 + ((\lg y - 3) / \lg 5)$$

$y < 1000$ 时：

$$x = 1$$

$y \geq 300000$ 时：

$x = 0$ 返回报错：输入文件过大，要求用户拆分文件（否则程序的摘要生成的处理时间会过长）。

把提取出的文字内容按照 1000 字为单位进行切片。

i 初始化为 0。当 $i < x$ 时：将每 1000 字切片自动投入 Ollama 上面的本地 Deepseek 小体量模型总结出 200 字第 i 级总结。如果切片总数大于 5 时，再将每 5 段 200 字总结合并产生的 1000 字切片自动投入 Ollama 上面的本地 Deepseek 小体量模型总结出 200 字 $i+1$ 级总结。最后，将不多于 5 的 200 字 x 级总结合并改写为 1000 字的最终总结。

摘要生成程序的运行界面在树节点分析过程中，按照已经完成的运算处理节点数量、和未完成的运算节点数量进行比值计算，得到系统完成运算的百分比，在用户界面显示出摘要生成的百分比，便于用户查看摘要生成进展。

表 4-1 本课题的 PDF 切片提取摘要并合并的运行结构

```

Begin
  └─ get PDF text → calculate PDF textfile length y

  └─ judge y:
    └─ y >= 300000 → get error and exit
    └─ y < 1000 → x=1
    └─ 1000 ≤ y < 300000 → x=1 + ((lg(y)-3)/lg5)

  └─ Initialize cut slices: cut slices as 1000 chars → slices

  └─ Initialize i=0, current_slices = slices

  └─ Process Loop:
    while i < x:
      | generate i-stage summaries

      | if len (summaries) >5:
        |   merge i+1 stages slices → current_slices
        |   i +=1
      | else:
        |   break

    └─ merge final summaries → get 1000-char summaries
End

```

本课题的 PDF 文件文字流核心后端处理逻辑为当 $i < x$ 时：建立当前总结。对当前切片的内容：生成切片长度为 200 的切片总结，并对子级总结递归到下一级子级总结。最后，将不多于 5 的 200 字 x 级总结合并改写为 1000 字的最终总结。

切片树算法与直接读取字符串相比，切片树对稳定性和性能的提升的实际运行的性能实验数据，参见后文第 5 章第 5.2 节对模型切片处理的本地运行测试结果数据。

表 4-2 本课题的 PDF 切片提取摘要核心后端处理逻辑：

```

while i < x:
    summaries = [] // generate current stage summaries
    for slice in current_slices:
        summary = OLLama.Deepseek.generate(slice, length=200)
        summaries.append(summary)
        if len(summaries) > 5: // judge if go to next stage
            new_slices = [] // merge each 5 summaries as a new stage Slice-Tree
            for j in range(0, len(summaries), 5):
                merged_slice = "".join(summaries[j:j+5])
                new_slices.append(merged_slice)
            current_slices = new_slices
            i += 1
    else:
        break // End summaries generation

```

本课题研究的 PDF 切片提取摘要树 Slice-Tree 的内核简化运行流程如图 3-1：

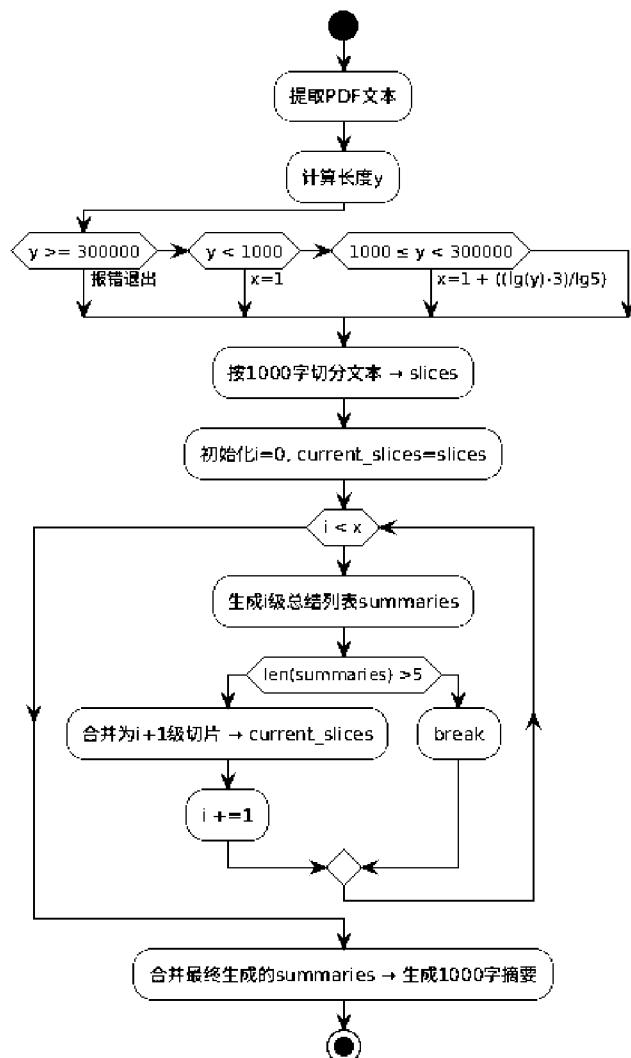


图 4-1 本课题研究的 PDF 切片提取摘要树 Slice-Tree 的内核简化运行流程

4.2 PDF 文档格式与 PDF 文字和图表读取处理接口

4.2.1 PDF 文档的整体格式

PDF 文档全称为便携文档格式，是 Adobe 公司于 1994 年发表的文档格式，Adobe 公司在 2008 年在 ISO 国际标准化组织公开确认了 PDF 格式标准^[6]。PDF 文档使用文字、点阵图、矢量图三种形式存储文档。PDF 具有不易修改、格式在不同设备上面具有一致性的优点。

PDF 文件分为 5 部分^[6]：文件标识符、文件结构控制符、文件结构标识符、PDF 数据流、xref 交叉引用表 5 部分。

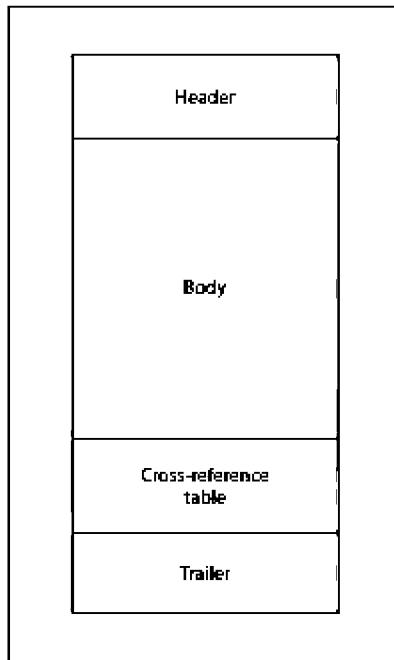


图 4-2 PDF 文件组成

PDF1.7 的文件头为%PDF-1.7\n。PDF 文档内容数据流通过分隔符隔离，分隔符开头为数字表示唯一数据流的序号。分隔符开头第 2 个数字表示数据流被修改次数。obj\n 表示数据开始。endobj\n 表示本段数据流终止。

PDF 文件结尾前一段为交叉引用表 xeof，最后一段%%EOF 为 PDF 文件结尾。PDF 交叉引用表用于 PDF 文件修改后保存。PDF 文件通常在修改时在文件结尾增加修改后增加的内容，在文件修改删除处删除修改时删除的内容。

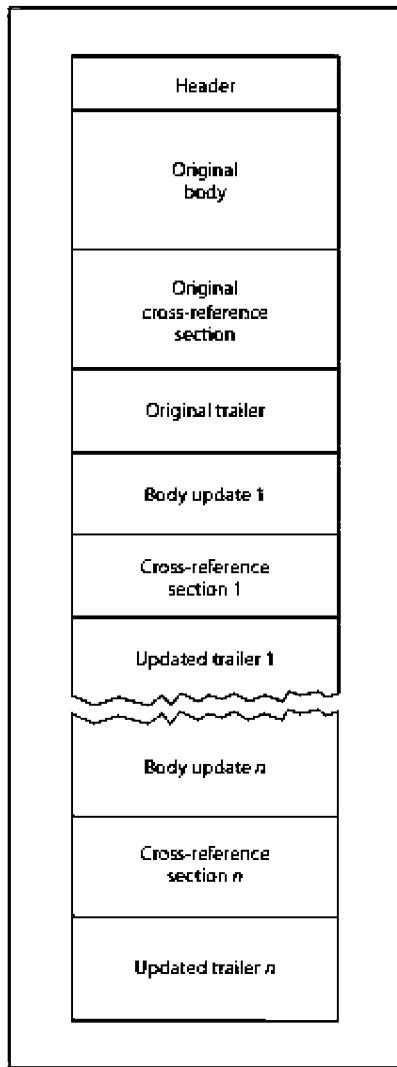


图 4-3 PDF 文件修改后的结构

PDF 实际文件有明文简单情况，和多种二进制数据流的复杂情况。较为简单的情况明文型 PDF 如表 4-3 所示。明文型简单 PDF 使用明文数据流，因此明文数据流 PDF 可以直接从 stream 数据流里面明文导出来。

4.2.2 PDF 文件的整体读取方法

表 4-3 所述的明文型 PDF 文件头为：%PDF-1.7，第一个对象 1 0 obj 为分页序号信息分为类型、页数和页面细节标注对象位置 Kids，以 endobj 结束。第二个对象 2 0 obj 为页面尺寸数据，它的信息为页面尺寸 MediaBox、边框的尺寸信息，标注页面细节的母对象位置 Parent 和资源库对象 Resources、内容对象 Content。第三个对象 3 0 obj 为文本框字体资源库 Resources 信息，这里设定的字体 F0 为 Times-Roman 字体，设定字体子类型为 Type1 型。第四个对象 4 0 obj 为页面数据流内容信息，标注长度 Length 为 65

字节，内设数据流，其中数据流文字 Tj 为： PDF case file, 字体为预先定义的 F0 字体，数据流还标注了数据流文字的字号 36 和位置格式 Tf, cm 表示按照平移变换显示文字，
1. 0. 0. 1. 50. 700. cm 表示将当前坐标系的原点平移到(50, 700)。第五个对象为页面根节点 Root，它的类型为 Catalog 目录类型，确认起始页面的位置为对象 1，R 表示根节点 Root 的分隔符，根节点还可以分为子根节点，遍历完所有的页面子根节点后才可以得知 PDF 的页面数量。

交叉索引表 xref 首先写明起始位置 0 号节点（文件头），节点数量为 6（含 0 号节点），随后进入交叉引用表正文。正文第一个引用表为例行表起始节点 0 号节点默认为 0000000000 65535 f，其中 65535 为生成号，f 为表示无效的状态标记。生成号表示删除次数，除 0 节点以外生成号默认为 0 否则一些编辑器会报错。1 号节点为 0000000010 00000 n，n 表示有效状态的状态标记，生成号 0 表示未经过修改，10 表示 1 号节点开始于文件第 10 字节。其后为 trailer 表示根节点位置为 5 号节点，总共有 6 个节点。最后 startxref 表示交叉索引表起始位置为 500，文件结尾是 EOF。

当 PDF 阅读器读取文件时，首先第 1 步读取文件结尾 EOF 之前的 startxref 交叉索引表起始位置。第 2 步转到交叉索引表 xref 读取每个节点的节点号并且记录下来。第 3 步读取根节点作为首页，随后根据根节点子根节点的页面信息确定页面数量和每一页的顺序，最后根据页面内容节点 content 记录的流内容渲染页面内容。

表 4.3 明文型简单 PDF1.7 文件内容示例

%PDF-1.7	Stream
1 0 obj	1. 0. 0. 1. 50. 700. cm
<< /Type /Pages	BT
/Count 1	/F0 36. Tf
/Kids [2 0 R]	(PDF case file) Tj
>>	ET
endobj	endstream
2 0 obj	endobj
<< /Type /Page	5 0 obj
/MediaBox [0 0 612 792]	<< /Type /Catalog
/Resources 3 0 R	/Pages 1 0 R
/Parent 1 0 R	>>
/Contents [4 0 R]	endobj
>>	xref
endobj	0 6
3 0 obj	0000000000 65535 f
<< /Font	0000000010 00000 n
<< /F0	0000000076 00000 n
<< /Type /Font	0000000199 00000 n
/BaseFont /Times-Roman	0000000335 00000 n
/Subtype /Type1 >>	0000000447 00000 n
>>	trailer
>>	<< /Size 6
endobj	/Root 5 0 R
4 0 obj	>>
<</Length 65>>	startxref
	500
	%%EOF

4.2.3 PDF 文件对长数据流的二进制压缩存储和读取方式

对复杂类型的 PDF 文件，通常有图片或图表数据流对象，和二进制压缩编码流两种流类型。其中直接的文字类文档 PDF 也可以把文字放入二进制压缩编码流里面。

PDF 文档的二进制流式压缩原理和示例如下。在表 4-4 右侧栏示例里面的二进制压缩编码型数据流的典型、常见的二进制编码压缩格式是 FlateDecode，流内长度为 76744

个字符。FlateDecode 使用的是 zip 文件的哈夫曼树状编码压缩方式，部分的 PDF 文档对部分数据流加密也是通过二进制压缩编码过程加密实现的。FlateDecode 文件流由这几部分构成：第 0 位 bit 为 bfinal，认证是否为最后一块 0 为否 1 为是；第 1-2 位数为 btype，00 表示无压缩，01 表示静态哈夫曼树，10 表示动态哈夫曼树，11 表示无意义或文件损坏。00 无压缩块内容为：16 位数的 LEN 表示小端字节序的块数据长度，16 位数的 NLEN 表示 LEN 的按位取反校验码，其后为 LEN 字节长度的原始数据。

01 模式通常用于存储 ASC-II 简易英语字符串数据，01 静态哈夫曼压缩块的内容为：先使用 LZ77 算法进行字符串压缩，FlateDecode 系统 01 模式下预定义一个内置的静态哈夫曼树。01 模式的 LZ77 算法压缩方式为：字面量 Literal、结束标记 End-of-Block、长度代码 Length；字面量为 0-255 的二进制数，结束标记为 256 二进制数，长度代码 L 为：长度范围代码 $E_c +$ 额外位数 E_b ， $L = E_c - B + E_b$ ，其中 B 是长度范围代码基数（B 是程序内置的常数， E_c 和 E_b 是存储内容）。257-285 为长度范围代码，长度范围代码基数 B 和 E_b 取值范围为：B = 254， $E_c = 257\sim264$ ， $E_b = 0$ ；B = 254， $E_c = 265\sim268$ ， $E_b = 0\sim1$ ；B = 250， $E_c = 269\sim272$ ， $E_b = 0\sim3$ ；B = 238， $E_c = 273\sim276$ ， $E_b = 0\sim7$ ；B = 210， $E_c = 277\sim280$ ， $E_b = 0\sim15$ ；B = 150， $E_c = 281\sim284$ ， $E_b = 0\sim31$ ；B = 27， $E_c = 285$ ， $E_b = 0$ 。01 模式在 LZ77 算法后，使用哈夫曼编码树表示 LZ77 编码结果的编码方式是：0-143 为 8 位二进制数使用 00110000 – 10111111 二进制表示；144-255 为 9 位二进制数使用 110010000 – 111111111 表示；256-279 为 7 位二进制数使用 0000000 – 0010111 表示；280-287 为 8 位二进制数使用 11000000 – 11000111 表示。

10 模式为动态哈夫曼编码二叉树。10 模式压缩块内容为：动态哈夫曼树常量编码表(C_t) + 01 模式的 LZ77 压缩结果的动态编码哈夫曼树 + 分块终结符 EndofBlock。01 模式动态哈夫曼树常量编码表存储方式为： $C_t = 5$ 位 bit 的 HLIT 码 + 5 位 bit 的 HDIST 码 + 4 位 bit 的 HCLEN 码 + $(HCLEN + 4) \times 3$ (3 bit 位数编码) 的 HCLEN 码表 T + 常量对应码表 T_c 。HLIT 码是 257-286 与 10 模式的长度范围代码 $HLIT = E_c + 257$ 取值范围一致。HDIST 码是表示二进制位数-1 也就是 bit-1，HDIST 码的取值范围是 0-31 与 10 模式长度范围代码的额外位数 $HDIST = E_b$ 一致。HCLEN 表示码表位数，4bit 的 HCLEN 数字是从 4-19，4-19 为码 L 表的前 4-19 个数。HCLEN 码的码长表自身的长度编码 HCLEN 取值 4-19，HCLEN 码表内容码表长表示 LZ77 算法里面的子码表 E_b 的额外位数的长度，HCLEN 表消耗的存储空间为 $(HCLEN + 4) \times 3$ (3 bit 位数编码)。HCLEN

码长表的数字含义表 T 按照顺序 0-19 依次为 16, 17, 18, 0, 8, 7, 9, 6, 10, 5, 11, 4, 12, 3, 13, 2, 14, 1, 15, HCLEN 数字对应码表前 HCLEN + 4 个码表数值子表。这些码长数 L 以 3 位 bit 整数 0-7 表示，如上所述 0 长度编码对应符号为空/不使用。码长数 L 为 0-15 表示长度 0-15；16 表示重复之前的数字码（code）3-6 次，下 2 位 l_e 表示长度 $L = l_e + 3$ ；17 表示重复之前的数字码 3-10 次，下两位 1 表示长度 $L = l_e + 3$ ；18 表示重复之前的数字码 11-138 次，下两位 1 表示长度 $L = l_e + 11$ 。符号码长量表 HCLEN 为动态哈夫曼树码长。HCLEN 子表的应用实例如下：当符号码长量为 [8, 8, 8, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0] 时，符号取值为这几个数：16, 18, 8 需确保码长编码表覆盖所有可能使用的符号，数据中后续有其他符号（如码长 8），需包含符号 8，因此这种情况下 HCLEN 必须是一个大于 $5 - 4 = 1$ 的数（HCLEN = 1~15 在这个案例里面都是正确的）。HCLEN 后面的码表使用 3 位数表示数字含义表 T 常量码长为 0-7 码长（也就是码长 1-8 位二进制数 bit），表按照顺序标出常量码对应的码长，表顺序为常量码对应的码。按照码长数字含义表 T 的空间读取 HLIT 次，从而读取常量对应码表 T_c 。根据常量对应码表 T_c 翻译哈夫曼树还原为 LZ77 压缩数据，从而解算出原文件流数值。

4.2.4 PDF 文件对图片和表格数据流的存储和读取方式

PDF 文档图片型数据流存储原理如下。表 6 左侧栏(1)示例里面的图片型数据流。常见的 PDF 图片数据是明文位图 BMP 格式，BMP 使用未经压缩的二进制明文表示每个像素的 RGB 或者灰度等颜色，从而显示出位图图片。表 6 的左侧栏(1)示例里面的明文状态的每个像素 BitsPerComponent 使用 8 字节表示，颜色空间 ColorSpace 是设备灰颜色空间，示例里面的 PDF 流节点使用 FlateDecode 压缩图像编码，高度为 3271，类型为 XType 插入类型，图片流长度为 16146 字节。

这一段介绍 PDF 文档表格型数据流存储原理如下。表 6 左侧栏(2)示例里面的表格型数据流，3 w 0 0 0 RG 10 100 m 200 100 1 S 流语句是向量绘图语句，这个流语句是表示绘制从(10,100)移动到初始化(m 意思是 moveto)到(200,100)的直线型横线 1，结束符 S 指的是描边(Stroke)；50 80 m 50 120 1 S 流语句表示绘制从(50,80)到(50,120)的向量竖线，通过字符串的位置排列、同时绘制横竖线向量图构建表格。PDF 的向量直线绘图还可以设置线宽 w 和颜色 RG，例如 3 w 可以设置线宽度为 3 单位（默认线宽度为 1 单位），颜色 RG 的设置是三位数字(0~1)分别表示红色、绿色和蓝色的亮度，默认黑色的线条颜

色为 0 0 0 RG 表示红、绿、蓝亮度为 0 时就显示出来黑色的线条。完整的表示线宽的实例先记录向量线宽，第 2 步记录颜色，第 3 步记录起点，最后记录终点和类型。

表 4-4 复杂型 PDF1.7 文件对象流内容示例

复杂图片图表型 PDF1.7 文件对象流示例	二进制压缩编码型 PDF1.7 文件对象流示例
<p>(1) 图片型对象流：</p> <pre> 16 0 obj << /BitsPerComponent 8 /ColorSpace /DeviceGray /Filter /FlateDecode /Height 3271 /Subtype /Image /Type /XObject /Width 5075 /Length 16146 >> stream %此处省略 16146 个二进制图片数据流字符 endstream endobj </pre> <p>(2) 表格型对象流：</p> <pre> 253 0 obj << /BBox [0 0 1128.39856892 668.646125] /FormType 1 /PTEX.FileName (./figures/dsr1_performance.PDF) /PTEX.InfoDict 588 0 R /PTEX.PageNumber 1 /Resources << /ExtGState << /A1 << /CA 0 /Type /ExtGState /ca 1 >> /A2 << /CA 1 /Type /ExtGState /ca 1 >> /A3 << /CA 0.9 /Type /ExtGState /ca 1 >> /A4 << /CA 0.8 /Type /ExtGState /ca 0.8 >> >> /Font << /F1 589 0 R /F2 590 0 R /F3 591 0 R /F4 592 0 R >> /Pattern << /H1 257 0 R >> /ProcSet [/PDF /Text /ImageB /ImageC /ImageI] /Shading << >> /XObject << >> >> /Subtype /Form /Type /XObject /Length 2958 >> stream 3 w 0 0 0 RG 10 100 m 200 100 1 S %画横线 50 80 m 50 120 1 S %画竖线 S 指描边(Stroke) %此处省略二进制数据，流大小总计 2958Byte endstream endobj </pre>	<pre> 8 0 obj << /Filter /FlateDecode /Length 76744 /Length1 355832 /Type /Stream >> stream %此处省略 76744 个二进制压缩编码字符 endstream endobj </pre>

4.2.5 以开源接口改编的 PDF 文件的文字和表格内容提取后端

PDF 文件在字节流结构部分记录文字和图片，在 stream 流内设置图片和文字、表格的矢量图形向量，PDF 直接使用字节流文档形式而不是以内附成型图片形式表示的文字和图表，通常可以直接通过程序以相对较高的效率读出。

由前文 3.4 指出了本课题系统后端的其余模块的架构和运行处理方式。本小节介绍系统文字和表格提取后端的处理架构和运行方式。

由前文 4.2.2 指出了 PDF 文档的文字信息是坐标信息、字体信息和文字信息组成的字节流构成的。由前文 4.2.4 指出了 PDF 文件的表格内容是带有坐标的文字信息和坐标向量的矢量图信息混合形成的，因此按照 PDF 文档的坐标信息提取文字信息的同时也会同步地提取所有非图片类型的 PDF 表格内含的表格内附的文字数据信息。

Npm-PDFreader 是一款开源的 PDF 文件处理接口，可以在 Visual Studio 这种 IDE 上面应用于 Visual Studio 的基于 node.js V15.14 的 vue 2 的 PDF 文件读取工作。安装完 Npm-PDFreader 以后，在基于 node.js V15.14 的 vue 2 的 vue 项目中添加 npm-PDFreader 依赖以后，可以使用 npm-PDFreader 文件处理接口读取处理 PDF 文件里面直接以字节流存储的文字。

根据前文 4.2.2 介绍的 PDF 文件详细存储结构与读取方式，npm-PDFreader 这种处理工具读取 PDF 文档内含文字和表格信息的模块工作流程是：

1、直接读取未压缩的流模块 stream 文字及其明文坐标，或者根据前文 4.2.3 介绍的 FlatDecode 代表的长数据流二进制压缩读取方式解压缩流，随后解压后读出 PDF 文档内部所有显示出的有效流 stream 模块的文字流信息，记录流内的每一个文字的坐标。

2、按照文字流和坐标所在的页面顺序为基准，先对文字流按照页面顺序从前往后进行排序。

3、按照文字流的明文坐标顺序，按照从页面左侧往页面右侧坐标读取排序顺序、按照从页面上侧到页面下侧的坐标顺序，在第 2 步对文字流按照页面顺序进行排序的基础上，进一步去对页面内部的文字流进行排序，并输出文字流排序结果。

第 3 步输出的文字流排序结果，将坐标和字号对比邻接的字符合并为一个单词，在坐标换行或者坐标和字号对比相邻字符有空位没有填充字符的位置，均以空格填充字符串，产生和得到 PDF 文件提取文字和表格产生的插入了空格的长字符串 S。这个长字符串 S 就是本系统的 PDF 文件的文字和表格提取后端提取到的目标文字流。

表 4-5 本课题基于开源项目改编的 PDF 后端文本流读取改编的接口简化代码 PDFserve.js

```

import { toggle } from "../npm-PDFreader-3.0.1/lib/LOG.js";
import { PDFReader } from "../npm-PDFreader-3.0.1/index.js";
//const toggle = require("../npm-PDFreader-3.0.1/lib/LOG.js");
//const PDFReader = require("../npm-PDFreader-3.0.1/index.js");
toggle(false);

/*exports.*/
async function FileItemsPromise(filename){
    return new Promise((resolve, reject) => {
        let content = "";
        new PDFReader().parseFileItems(filename, (err, item) => {
            if (err) return reject(err);
            else if (!item) return resolve(content.trim());
            else if (item.text) content = content + item.text + " ";
        });
    });
}

var filename = process.argv[2];
if (!filename) {
    console.error("please provide the name of a PDF file");
} else {
    console.log("", filename, ".");
    const content = await FileItemsPromise(filename);
    console.log("", content);
}
}

```

PDF 文档文字流运行读取实例：

供 PDF 读取后端读取的 PDF 实例文档内容如下：

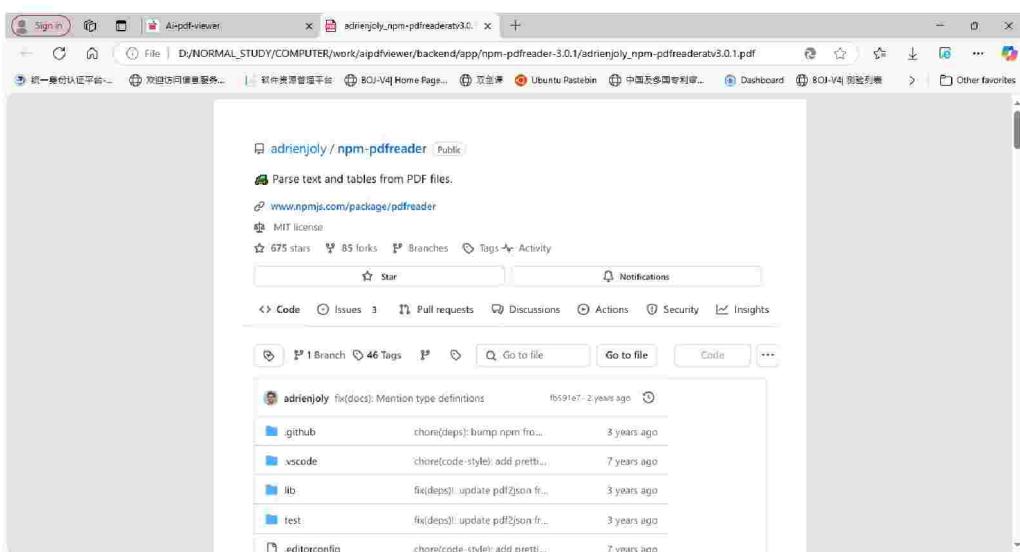


图 4-4 示例 PDF 文档内容

本课题后端使用改编的开源程序接口 `npm-PDFreader` 读出示例文档文字流字符串的读出效果是把 PDF 内文字和表格输出为长字符串形式。

表 4-6 本课题后端使用开源程序接口 `npm-PDFreader` 读出示例文档文字流读出效果

```
D:\NORMAL_STUDY\COMPUTER\work\aiPDFviewer\backend\app\npm-PDFreader-3.0.1>node
PDFserve.js adrienjoly_npm-PDFreaderatv3.0.1.PDF
printing raw items from file: adrienjoly_npm-PDFreaderatv3.0.1.PDF ...
getOut: adrienjoly / npm-PDFreader Public  Parse text and tables from PDF files. www.npmjs.com/p
ackage/PDFreader MIT license 675 stars 85 forks Branches Tags Activity 1 Branch 46 Tags Go
to file Go to file Code adrienjoly fix(docs): Mention type definitions fb591e7 · 2 years ago
//此处省略一些文字输出结果
.then(displayTable), Rule.on(/^Values\:/).accumulateAfterHeading()
.then(displayValue), ]); new PdfReader().parseFileItems("test/sample.pdf", (err, item) => { if (err) console.error(err); else processItem(item); });
}); Troubleshooting & FAQ Is it possible to parse a PDF document from a web application?
Dmitry found out that you may need to run these instructions before including the pdfreader module:
Releases 44 v3.0.7 Latest on Jan 22 + 43 releases Sponsor this project https://adrienjoly.com/donate/
Contributors 23 + 9 contributors Languages HTML 93.8% JavaScript 5.5% Rich Text Format 0.7%
Cannot read property 'userAgent' of undefined err or from an express-based node.js app global.navigator={userAgent:"node",
window.navigator={userAgent:"node",
```

本课题系统的 PDF 读取后端在这个读取模块的基础上，构建了一个端口号为 8012 的本地 PDF 文字和表格 node 的 js 读取服务接口 api，通过 bat 脚本启动这个 PDF 读取后端服务。这个 PDF 读取后端的工作时序是这样的：

- 1、Vue 前端接收文件后，向 node.js 读取服务接口 api 传输文件内容。
 - 2、node 的 js 读取服务接口 api 创建 PDF 临时文件。
 - 3、node.js 的 PDF 读取后端在临时文件创建成功以后，把临时文件目录传递给 PDF 函数读取接口。
 - 4、node.js 的 PDF 读取后端内部的读取函数接口按照上述的算法读取 PDF 文件的文字和表格内容，产生长字符串 S。
 - 5、node.js 的 PDF 读取后端产生的长字符串 S，通过 api 接口传输，返回传输给 vue 前端。
 - 6、回传成功以后，node.js 的 PDF 读取后端删除用于接口读取的 PDF 临时文件。
- 综上所述，本 PDF 读取后端的文件处理时序图如图 4-5 所示。

PDF文件处理时序图 (端口:8012)

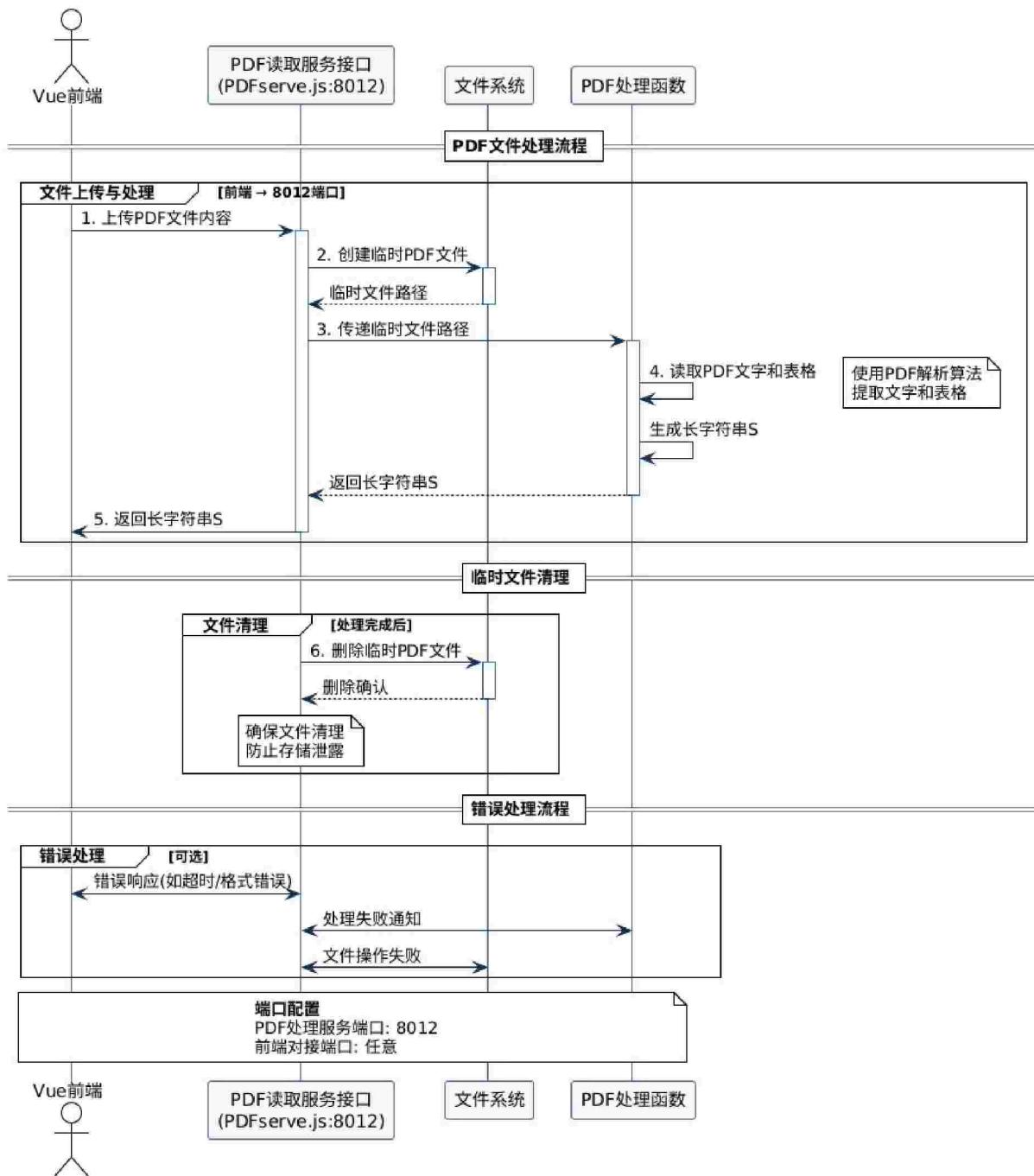


图 4-5 PDF 文档读取后端的系统时序设计

4.3 本章的内容小结

本章节介绍了 PDF 智能摘要生成系统的前后端核心处理接口的算法需求、前后端核心算法的目标原理和前后端核心算法的算法运行方法，同时介绍了系统后端核心的 PDF 文件的文字和表格读取后端的设计结构和 api 相关操作执行时序。

本章第 4.1 节部分分析了系统前端核心的文件处理接口算法 Slice-Tree 的算法需求、算法的目标原则、算法的算法方法结构。关于本章第 4.1 节切片树接口算法对性能的提升和切片窗口大小的选取数据，参见后文第 5 章的 5.2 节模型的本地运行模型运行信息遗漏率的程序测试和稳定性实验测试，选取合适的切片长度为 1000 字切片可以兼顾模型的低信息遗漏率和较高的本地运行输出稳定性。

本章第 4.2 节部分分析了系统后端核心的 PDF 文件的文字和表格读取的算法需求、算法的目标原则、算法的算法方法结构，最后本章第 4.2.5 节介绍了系统后端核心的 PDF 文件的文字和表格读取后端本课题专门改编的服务 api 相关文件系统操作后端时序设计。

第五章 系统的运行效果和测试评估

5.1 系统的运行效果

5.1.1 系统的软件运行环境和硬件配置

1. 系统的安装与运行环境是开源平台 OLLama V0.6.0 安装的本地 deepseek-r1:1.5b 开源模型、使用 node-js V15.14.0 开源的运行平台、mongoDB V5.0.30 开源数据库，搭配的开源 PDF 文字提取改装系统的原始版本为：npm-PDFReader V3.0.1。

系统的其他小微开源模块安装包已经通过 npm 执行 package.json 在本地完成了安装，直接拷贝整个文件夹即可自行本地复制和使用子模块到其他主机上离线运行。

系统使用 bat 脚本启动 Node 数据库后端、OLLama 后端、PDF 文件读取后端服务的启动 bat 脚本命令行：

```
set NODE_OPTIONS=
start .\backend\startback.bat
start .\oLlamaserve.bat
start .\backendpdfreader\startpdfreader.bat
npm run serve
```

2. 本机的服务运行硬件配置：2018 年生产的旧型笔记本低端娱乐显卡 NVIDIA GeForce 940MX，GPU 专用内存 2.9GB，GPU 共享内存 3.9GB，整机的硬件内存总计为 7.9GB。需要更低端的运行条件下，本课题可以降低模型自身内存申请为 400MB、以纯 CPU 模式 OLLAMA_NUM_GPU = 0、最高处理字符数量 OLLAMA_MAX_CTX = 4096 字符的方式降低速度运行模型。

本机的显卡相对老旧、低端，因此显卡模式和纯 CPU 模式在本机运行条件下差别不大，体现出纯 CPU 模式具备较高的运行适配性和对低档次硬件的运行兼容性。

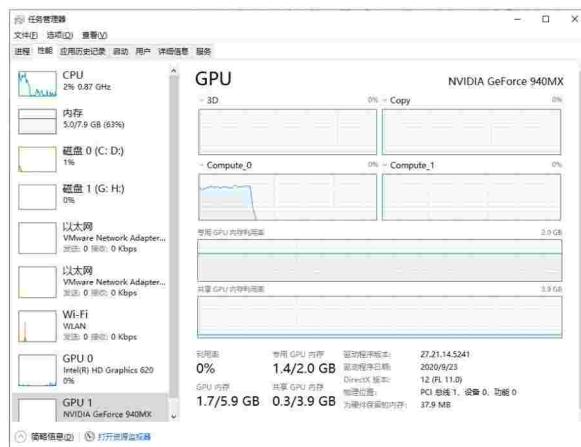


图 5-1 本机的硬件运行配置

5.1.2 系统的前端运行效果

1. 前端启动后的窗口如图 5-2。前端启动后设置的主页地址为 <http://localhost:8008/> 或者 <http://10.29.155.167:8008/> 的主页地址；这就是正常启动前端的画面。

```

C:\WINDOWS\system32\cmd.exe
WARNING: Compiled with 1 warnings
warning in ./src/components/Login.vue?vue&type=script&lang=js&
"export 'EventBus' was not found in '../main'

App running at:
- Local: http://localhost:8008/
- Network: http://10.29.155.167:8008/

Note that the development build is not optimized.
To create a production build, run npm run build.

```

图 5-2 登陆 vue 2 前端启动后的窗口

2. 无需登陆的前端帮助页面运行效果。帮助页面显示该程序 AI PDF Viewer 的使用说明书、软件遵循 GNU GPL3.0 协议；软件的运行环境为开源平台 Ollama V0.6.0 安装的本地 deepseek-r1:1.5b 开源模型、使用 node-js V15.14.0 开源的运行平台、mongoDB V5.0.30 开源数据库，搭配的开源 PDF 文字提取改装系统的原始版本为：npm-PDFreader V3.0.1。

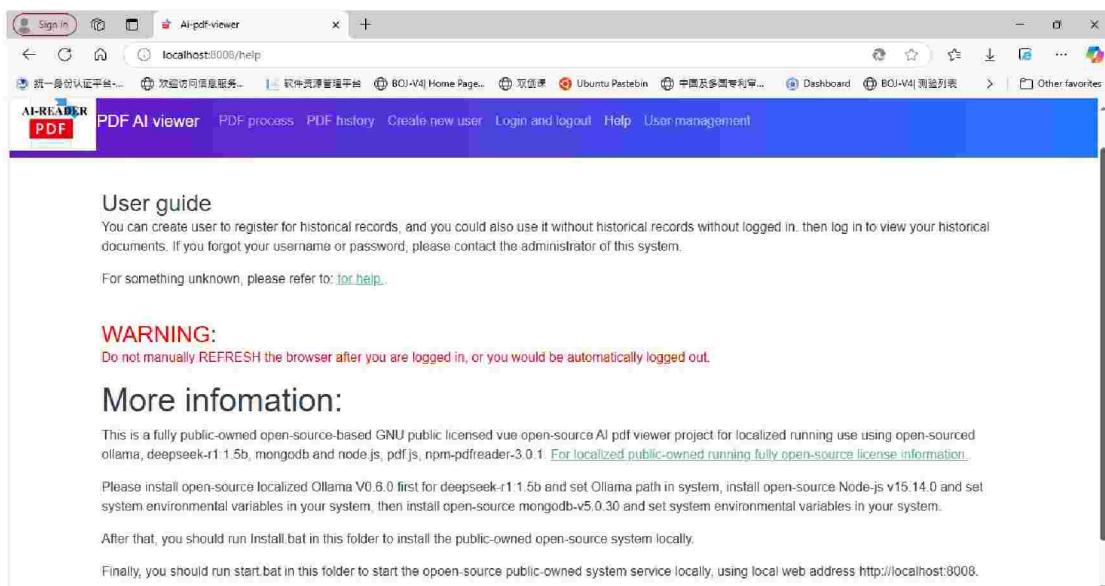


图 5-3 前端帮助页面运行效果

3. 登陆页面登陆前前端运行效果：登陆前登陆页面会提示用户在指定区域输入用户名和密码，菜单栏没有显示用户名。

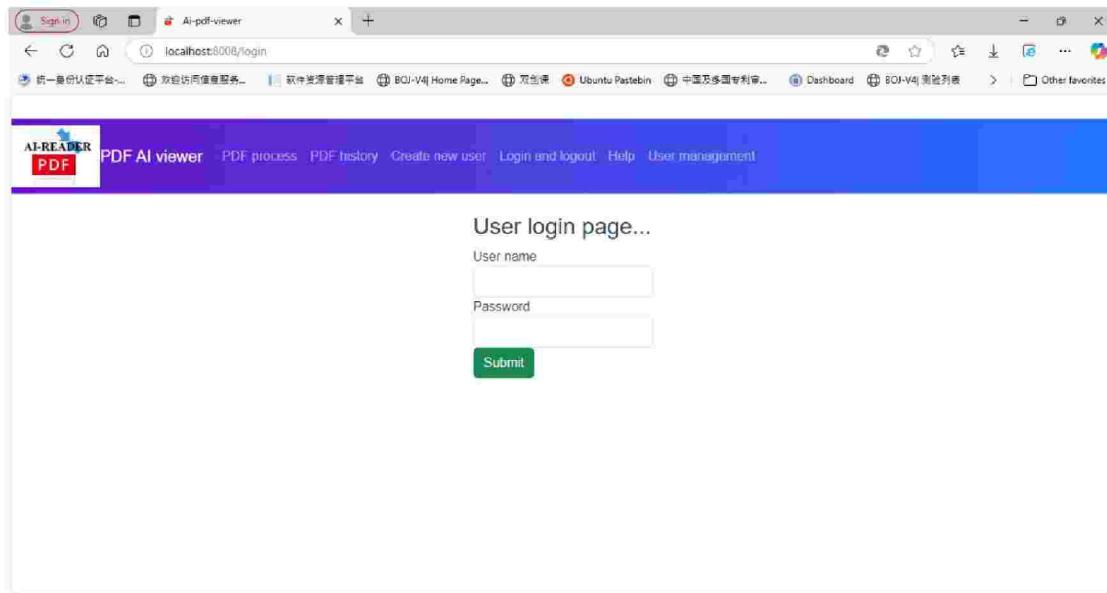


图 5-4 登陆页面登陆前前端运行效果

4. 登陆页面登陆后运行效果：菜单栏显示用户名，登陆界面显示成功登录。需要特别注意登陆后不要主动刷新浏览器，否则登陆脚本的内存会出现丢失。

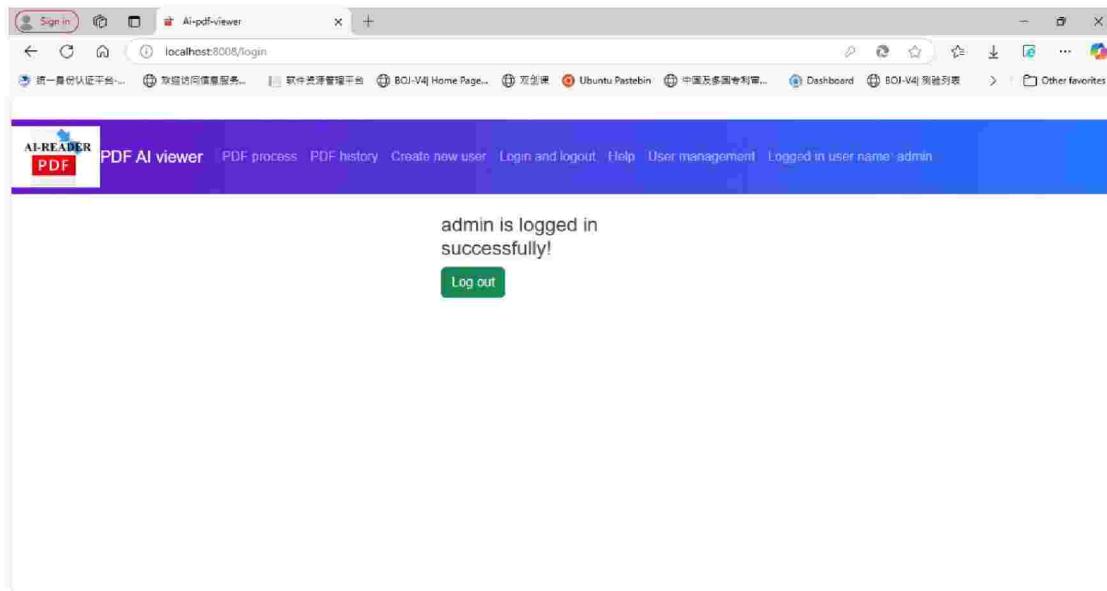


图 5-5 登陆页面登陆后的前端运行效果

5. 有权限的管理员登陆页面登陆后用户权限和密码管理页面运行效果。有权限的管理员可以管理全部的用户信息，编辑和删除全部的用户信息。其中管理员默认密码和默认账户是系统 Node 连接 MongoDB 数据库后端脚本自动默认设定的，就算误删管理员

账户，Node 连接数据库后端脚本也会补上管理员账户。

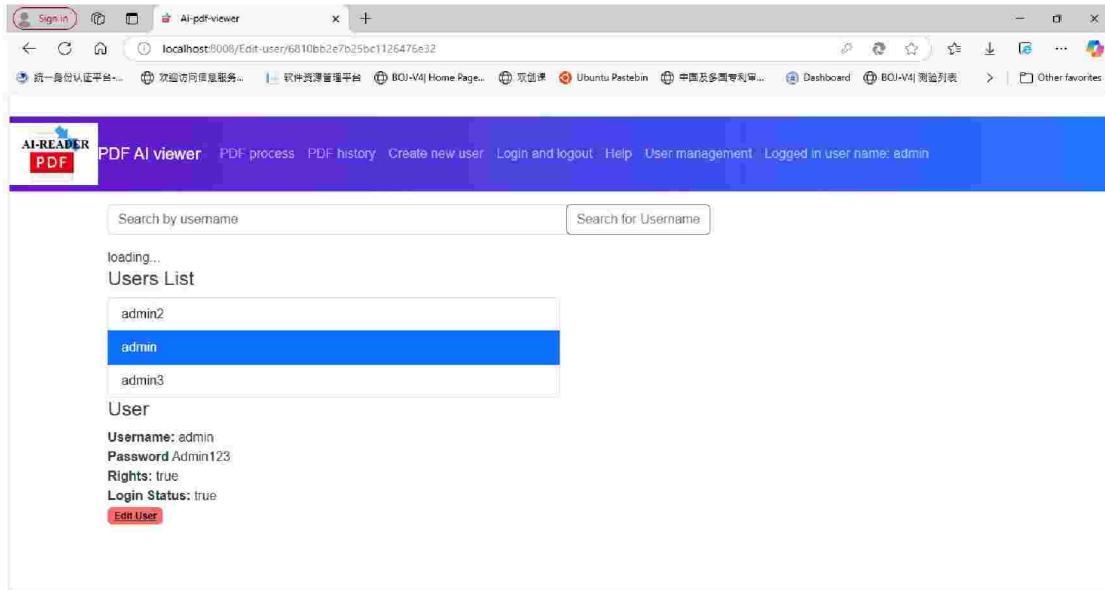


图 5-6 有权限的管理员登陆页面登陆后用户权限和密码管理页面运行效果

6.无权限的用户登陆后用户信息管理页面的效果。用户可以修改自己的用户名和密码（系统会审核保障修改后用户名和其他用户名不重复）。

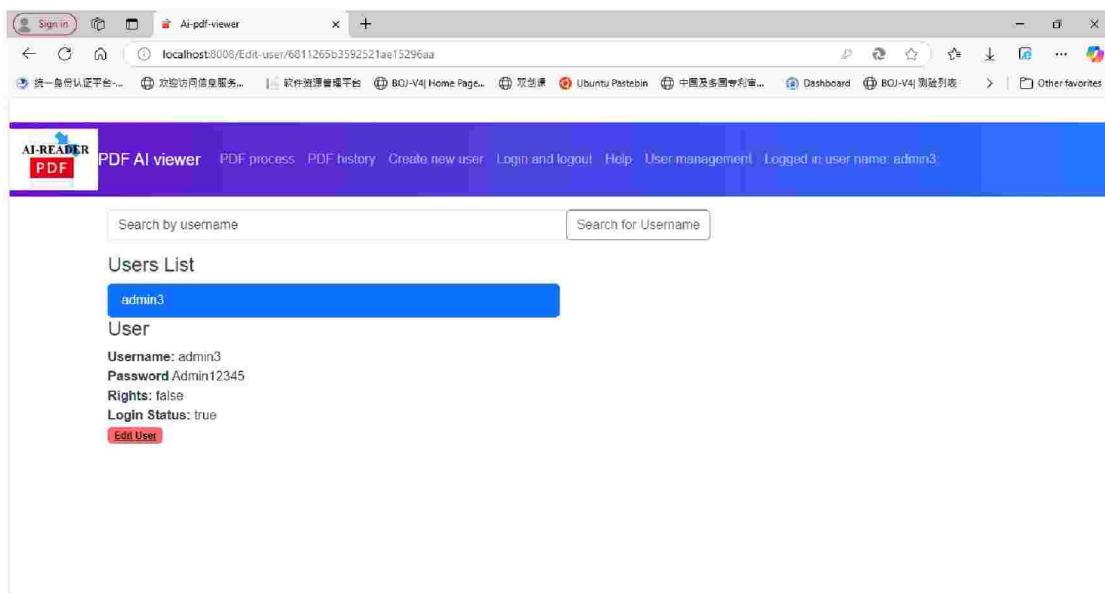


图 5-7 无权限的用户登陆后用户信息管理页面的效果

7.文件摘要生成系统前端可以为本地无历史记录模式的状态下不登陆状态的文件读取界面。上侧负责读取 TXT 文件，下侧负责读取 PDF 文件。下方与机器人的直接对话界面用于调试过程中主动测试本地人工智能模型运行状态，测试本地 oLlama 的 api 是否因为内存不足而退出的问题。

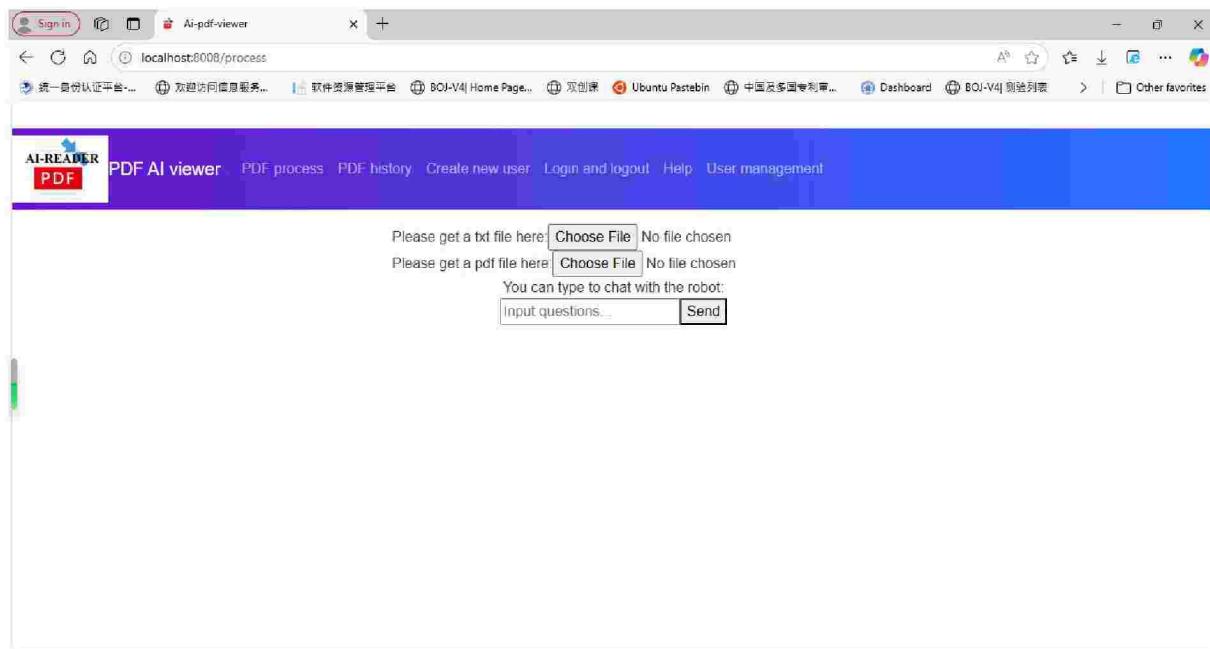


图 5-8 文件摘要生成系统前端可以为本地无历史记录模式的状态下不登陆状态的文件读取界面

5.1.3 系统的后端运行效果

1. MongoDB 数据库连接后端使用系统整体脚本打开的数据库后端自身启动脚本启动，同时需要启动 Mongoose 服务和 node 数据库连接后端自身的路由服务。

其中，node 服务的 `\backend\server.js` 是数据库连接后端的主服务。这里的 `\backend\setting.bat` 设置的 `D:\ProgramFiles\MongoDB\Server\6.0\data\db\test` 是 Mongoose 数据库在这个连接会话过程中使用的数据库服务数据存储地址。

```

.\backend\startback.bat
start .\backend\setting.bat
node .\backend\server.js

.\backend\setting.bat
mongod --dbpath D:\ProgramFiles\MongoDB\Server\6.0\data\db\test

```

MongoDB 数据库连接后端`\backend\server.js`的运行状态：MongoDB 数据库后端在 8080 端口运行，自动添加管理员账户：

```

D:\NORMAL_STUDY\COMPUTER\work\aiPDFviewer>node .\backend\server.js
Server is running on port 8080.
Connected to the database!
Admin has exist.

```

2. OLLama 后端在启动过程中，先配置本地 api 的服务本地地址，随后根据输入的参数修改端口号到 127.0.0.1:8011，根据参数修改申请的内存等空间，最后、加载 CUDA 的 GPU 仓库。执行过程中，如果正常连接但是内存申请不成功（内存不足或者显存不足是低配计算机运行 OLLama 时最常见的故障）则返回 HTTP500 代码；如果正常返回则返回 HTTP204 代码表示本地机器对话正常运行、处理了输入的数据。

本地 OLLama 后端的 token 词汇最大处理数量为 2048 个 token 词汇单位数。本地 OLLama 后端的运行状态如图 5-9。

```

C:\WINDOWS\system32\cmd.exe
2025/05/29 05:39:31 routes.go:1280: INFO server config env="map[CUDA_VISIBLE_DEVICES: GPU_DEVICE_ORDINAL: HIP_VISIBLE_DEVICES: HSA_OVERRIDE_GFX_VERSION: HTTPS_PROXY: HTTP_PROXY: NO_PROXY: OLLAMA_CONTEXT_LENGTH:2048 OLLAMA_DEBUG:false OLLAMA_GPU_OVERHEAD:0 OLLAMA_HOST:http://localhost:8011 OLLAMA_INTEL_GPU:false OLLAMA_KEEP_ALIVE:5ms OLLAMA_MAX_CACHE_SIZE: OLLAMA_MAX_LIBRARY: OLLAMA_LOAD_TIMEOUT:5ms OLLAMA_MAX_LOADED_MODELS:0 OLLAMA_MAX_QUEUE:512 OLLAMA_MODELSD:D:\ollama\NNODES OLLAMA_MULTIUSER_CACHE:false OLLAMA_NO_HISTORY:false OLLAMA_NO_PRUNE:false OLLAMA_NUM_PARALLEL:0 OLLAMA_ORIGINIS:!* http://localhost https://localhost http://127.0.0.1 https://127.0.0.1* https://0.0.0.0 https://0.0.0.0 http://0.0.0.0* https://0.0.0.0:app/* file:/# taunt://* vscode-webview://* OLLAMA_SCHED_SPREAD:false ROCR_VISIBLE_DEVICES:"]"
time=2025-05-29T05:39:31.855+08:00 level=INFO source=images.go:439 msg="total blocks: 5"
time=2025-05-29T05:39:31.856+08:00 level=INFO source=images.go:439 msg="total unused blobs removed: 0"
time=2025-05-29T05:39:31.866+08:00 level=INFO source=routes.go:1297 msg="Listening on 127.0.0.1:8011 (version 0.6.1)"
time=2025-05-29T05:39:31.869+08:00 level=INFO source=gpu.go:217 msg="looking for compatible GPUs"
time=2025-05-29T05:39:31.869+08:00 level=INFO source=cpu_windows.go:167 msg="packages count:1"
time=2025-05-29T05:39:31.869+08:00 level=INFO source=cpu_windows.go:214 msg="package=0 cores=2 efficiency=0 threads=4"
time=2025-05-29T05:39:31.869+08:00 level=INFO source=cpu.go:612 msg="Unable to load cudart library C:\windows\system32\nvcuda.dll: symbol lookup for cuCtxCreate_v3 failed: \xd\lx\bb\xbd\lx\bb\xd\xx\xd\xc\xd\xf\x2\x1\x\...\n"
time=2025-05-29T05:39:31.869+08:00 level=INFO source=cpu.go:612 msg="Unable to load cudart library C:\Windows\System32\nvcuda.dll: symbol lookup for cuCtxCreate_v3 failed: \xd\lx\bb\xbd\lx\bb\xd\xx\xd\xc\xd\xf\x2\x1\x\...\n"
time=2025-05-29T05:39:32.711+08:00 level=INFO source=types.go:130 msg="inference compute" id=GPU-4a2026f3-4d01-22d5-9931-728bb0eeca21 library=cuda variant=v11 compute=5.0 driver=0.0 name="" total=2.0 Gib available=1.7 Gib"
[GIN] 2025/05/29 - 05:39:47 [204] | 127.0.0.1 [OPTIONS] "/api/chat"
time=2025-05-29T05:39:47.810+08:00 level=WARN source=gnd.go:149 msg="key not found" key=qwen2.attention.block_count default=0
time=2025-05-29T05:39:47.815+08:00 level=WARN source=gnd.go:149 msg="key not found" key=qwen2.attention.key_length default=128
time=2025-05-29T05:39:47.815+08:00 level=WARN source=gnd.go:149 msg="key not found" key=qwen2.attention.value_length default=128
time=2025-05-29T05:39:47.815+08:00 level=WARN source=gnd.go:149 msg="key not found" key=qwen2.attention.block_count default=0
time=2025-05-29T05:39:47.817+08:00 level=WARN source=gnd.go:149 msg="key not found" key=qwen2.attention.key_length default=128
time=2025-05-29T05:39:47.817+08:00 level=WARN source=gnd.go:149 msg="key not found" key=qwen2.attention.value_length default=128
time=2025-05-29T05:39:47.818+08:00 level=WARN source=gnd.go:149 msg="key not found" key=qwen2.attention.block_count default=0
time=2025-05-29T05:39:47.819+08:00 level=WARN source=gnd.go:149 msg="key not found" key=qwen2.attention.key_length default=128
time=2025-05-29T05:39:47.819+08:00 level=WARN source=gnd.go:149 msg="key not found" key=qwen2.attention.value_length default=128
time=2025-05-29T05:39:47.820+08:00 level=WARN source=gnd.go:149 msg="key not found" key=qwen2.attention.block_count default=0
time=2025-05-29T05:39:47.821+08:00 level=WARN source=gnd.go:149 msg="key not found" key=qwen2.attention.key_length default=128
time=2025-05-29T05:39:47.822+08:00 level=WARN source=gnd.go:149 msg="key not found" key=qwen2.attention.value_length default=128
time=2025-05-29T05:39:47.904+08:00 level=INFO source=server.go:105 msg="system memory total=7.9 Gib free=5.2 Gib free_swap=5.0 Gib"
time=2025-05-29T05:39:47.904+08:00 level=WARN source=gnd.go:149 msg="key not found" key=qwen2.attention.block_count default=0
time=2025-05-29T05:39:47.905+08:00 level=WARN source=gnd.go:149 msg="key not found" key=qwen2.attention.key_length default=128
time=2025-05-29T05:39:47.905+08:00 level=WARN source=gnd.go:149 msg="key not found" key=qwen2.attention.value_length default=128
time=2025-05-29T05:39:47.908+08:00 level=INFO source=server.go:138 msg="offload library=cuda layers.requested=1 layers.model=25 layers.split="" memory_available=[1.7 Gib] memory.gpu_overhead=0.0 memory.required.full=1.9 Gib memory.required.partial=1.6 Gib memory.required.kv=56.0 Gib memory.required.allocations=[11.6 Gib] memory.weights.total=752.1 Gib memory.weights.repeating=752.1 Gib memory.weights.nonrepeating=182.6 Gib memory.graph.full=299.8 Gib memory.graph.partial=482.3 Gib"

```

图 5-9 OLLama 后端的运行状态

3.PDF 文件读取后端使用脚本启动，PDF 文件读取后端运行状态：PDF 读取后端在本地服务端口 8012 运行，和本地服务端口号为 8008 的 Vue 前端对接，自动添加临时文件夹和本地访问文件夹。Vue 前端通过 api 上传读取的 PDF 文件给 PDF 本地读取后端，PDF 本地读取后端把文件存入本地临时文件夹和本地访问文件夹。

```

D:\NORMAL_STUDY\COMPUTER\work\aipdfviewer>node ./backendpdfreader/pdfserve.js
backend service running on http://localhost:8012
configure connect with 8008 port
temp dir: D:\NORMAL_STUDY\COMPUTER\work\aipdfviewer\backendpdfreader\temp
local dir: D:\NORMAL_STUDY\COMPUTER\work\aipdfviewer\backendpdfreader\local

```

5.2 切片树接口 Slice-Tree 对系统准确性提升的效率评估

前文第 2 章从理论层面解度了机器学习模型如果输入越长的数据，越容易出现读取不均衡造成的信息遗漏问题，这是因为提高机器学习模型逻辑思维能力的准确性，这就会而导致机器学习读取信息遗漏的必然结果。

基于第 2 章所述的机器学习理论造成的读取长文本信息遗漏问题，前文第 4 章从算法层面介绍了解决机器学习模型读取长文本信息遗漏的切片树方法 Slice-Tree。在本章运行效率评估时，使用改变切片长度的方法，测试机器学习模型在读取不同长度文本时的关键词信息遗漏率，从而证明切片树接口 Slice-Tree 开源提高系统读取长文本的模型准确性。

设机器学习模型输入了长字符串 w ，通过 prompt 要求大模型输出输入数据内的检测到 j 个关键词 m_i 的语句片段。其中长度为 a_i 的关键词 m_i 出现了 n_i 次，则本地模型读取输入内容的^[27]信息遗漏率 E_j 是：

$$E_j = \left(1 - \frac{\sum_{i=1}^j (a_i \times k_i)}{\sum_{i=1}^j (a_i \times n_i)} \right) \times 100\% \quad \text{式 (5-1)}$$

其中， a_i 是第 i 个关键词的长度， n_i 是第 i 个关键词在文本内实际出现的次数， k_i 是第 i 个关键词在文本内被人工智能模型检测到的输出次数。

接下来设计配套的遗漏率评估 html 程序，根据式 5-1 计算模型输入不同长度文本时候，输出的遗漏率。

遗漏率评估 html 程序分为 2 部分：第 1 部分是输入样例生成程序，第 2 部分是模型的运行结果评估程序。

5.2.1 输入样例生成程序

输入样例生成程序包括：文本样例生成程序和关键词样例生成程序。

1、文本样例生成程序是一个字符串截取程序，输入长字符串 w ，字符串 w 的长度为 l ，输入截取长度 n ，设起始截取字符位置为 s

$$s = \lfloor (l - n) \times R \rfloor \quad \text{式(5-2)}$$

其中 R 是随机生成的 $0 \sim 1$ 之间的随机数 ($R < 1$)， $\lfloor \quad \rfloor$ 是向下取整函数确保样例不越界。则式 5-2 截取样例生成字符串范围是第 $s + 1$ 个字符 $w[s]$ 到第 $s + n + 1$ 个字符 $w[s+n]$ 结束的字符串 S_o 。

输入长字符串样 w 例来自本课题开题报告的 PDF 文字和表格在经过 PDF 读取后端提取后，产生的 11062 字的字符串，字符串切片长度范围设定为 300-5000。

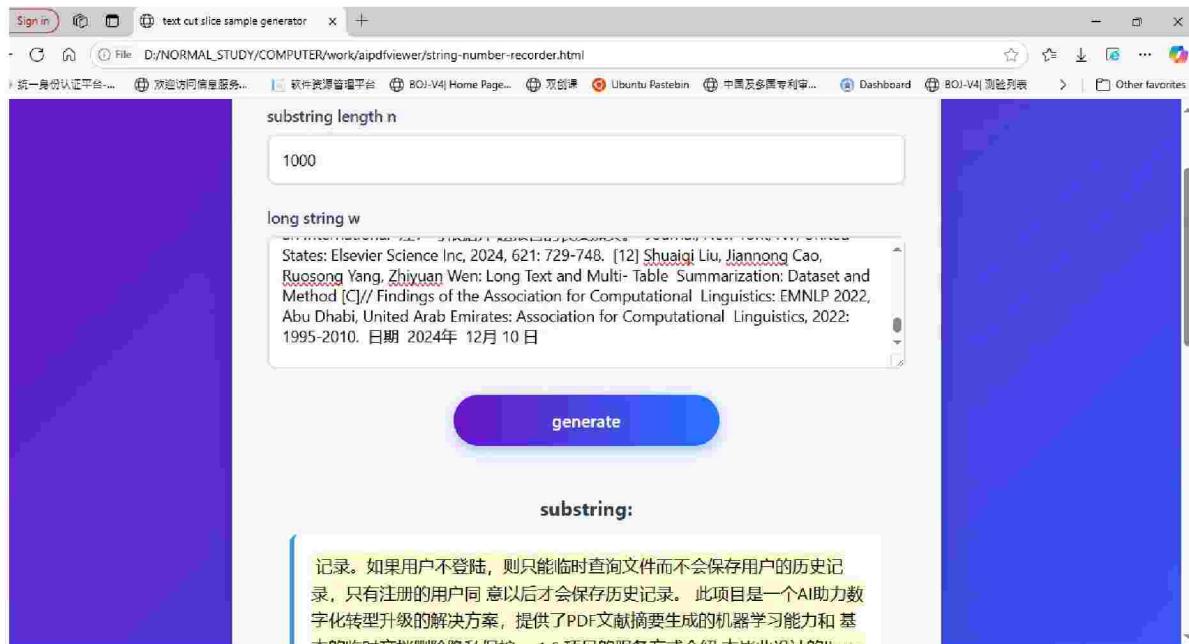


图 5-10 输入长字符串样例截取程序

2、关键词样例生成程序是一个字串提取程序，输入长字符串 w，提取前 n 个重复频率最高的字串 s，按照重复频率排序显示 n 个重复频率最高的子串。

使用字符串长度 2-16 的滑动窗口遍历算法，substr 遍历整个长字符串进行比对子字符串。比对时子字符串 substr 总是向后滑动窗口比对是否出现重复，比对出现重复后就把子字符串重复数量加入重复比对 Map 映射数据结构目录，记录字符串内容和比对重复次数。当后续 substr 发现之前 Map 映射数据结构目录已经存储了自己重复的次数时就不再比对，从而保障重复比对准确性和比对效率。

输入长字符串样 w 例来自本课题开题报告的 PDF 文字和表格在经过 PDF 读取后端提取后，产生的 11062 字的字符串，展示前 500 个高频重复字串。

使用关键词样例生成程序对样例进行字串词频分析，得到样例的高重复频率关键词列表：PDF/设计/服务/用户/文件/文档/系统/内容/接口/Llama/兼容。

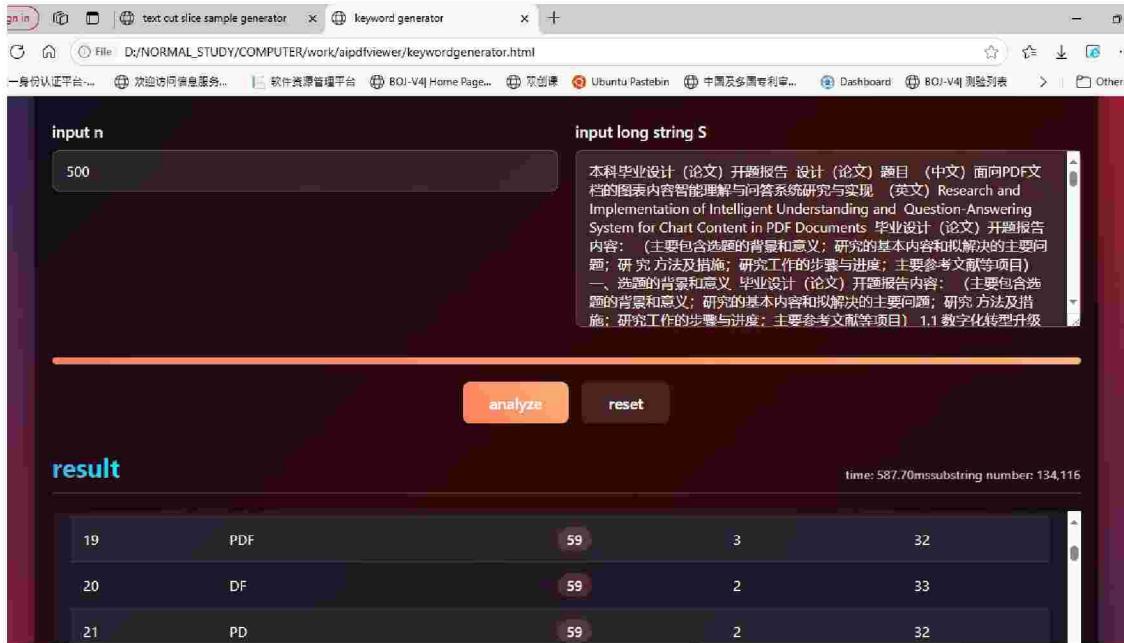


图 5-11 关键词样例生成程序

5.2.2 模型的运行结果评估程序

模型的运行结果评估程序输入文本样例生成程序产生的字符串长度为 n 的随机字符串 S_o 作为式 5-1 的变量输入字符串，输入关键词样例生成程序词频字串统计分析产生的关键词集合 $\{a_i\}$ ，它的计算目标是计算本章的信息遗漏率式 5-1 的遗漏率 E_j 。

该评估程序对接 Ollama 本地模型 api 的 8011 端口本地服务接口，使用 npm-chart.js 的图片 js 插件、外置 css 文本框界面插件实现图形界面。

评估程序通过读取字符串样例和关键词样例，直接使用字符串匹配搜索目标字符串 S_o 内部的关键词实际频率数量也就是式 5-1 里面的分母 $\sum_{i=1}^j (a_i \times n_i)$ 。

评估程序使用 prompt 向 Ollama 本地服务运行的本地语言模型输入字符串，在 prompt 提示语句内部规定输出规则，要求本地服务的语言模型输出返回任一含有输入关键词的输入字符串的句子节选。评估程序根据 Ollama 本地服务语言模型返回的字符串句子节选，计算模型读取的关键词数量也就是式 5-1 里面的分子 $\sum_{i=1}^j (a_i \times k_i)$ 。

本评估程序使用的 prompt 提问是：

**请逐句分析以下文本，不要遗漏任何信息，检查是否包含以下任何 1 个目标词：
 $\$\{targetPattern\}**\n$ 规则：1. 段落含有多个目标词，段落每出现 1 次任何 1 个目标词，输出含目标词的输入语句。2. 如果逐句完整阅读全文以后未找到目标词里面任何 1 个词汇，返回“未找到”3. 不要添加任何额外解释和目标词，只输出含目标词的输入语句/==== 文本 ====\$\{text\}==== 结束 ===/

其中， $\$\{targetPattern\}$ 是关键词的集合， $\$\{text\}$ 是输入样例的长文本内容。

随后评估程序执行式 5-1 的运算，得出信息遗漏率式 5-1 的遗漏率 E_j ，并通过图形化方式显示出来。

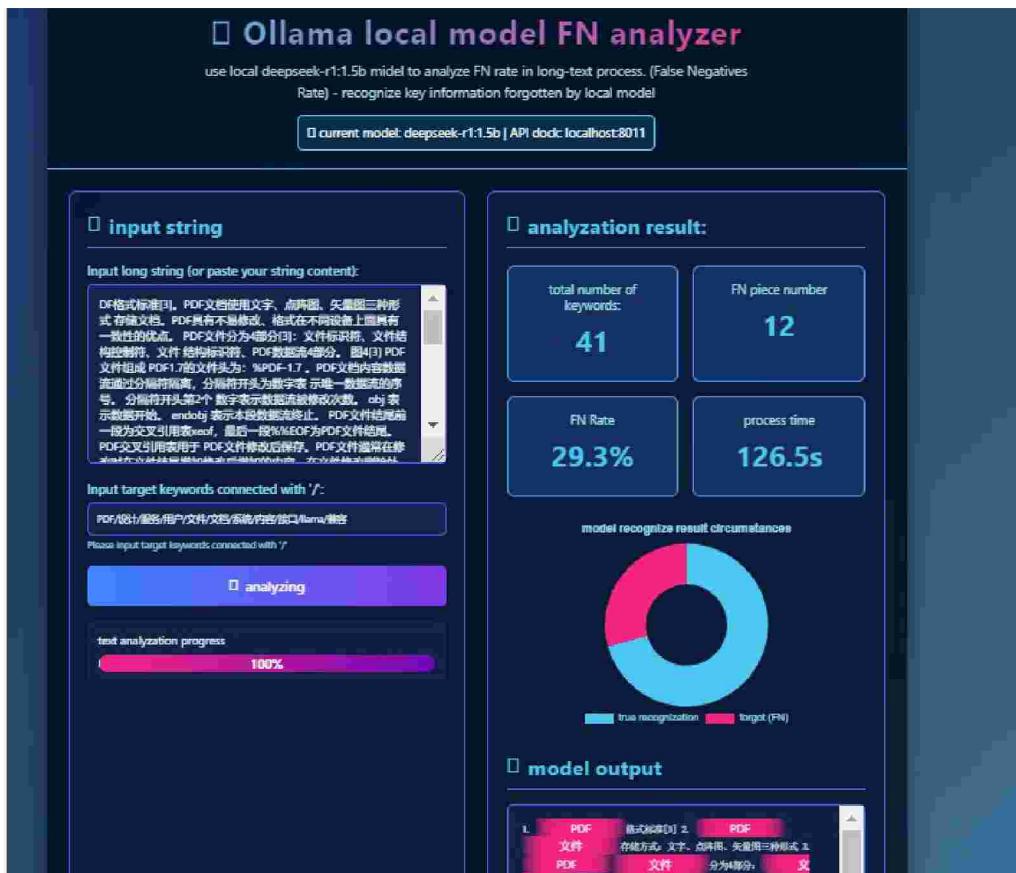


图 5-12 模型样运行结果评估程序

由于本机的配置不高，在本机实际运行的评估程序通常需要 50~150 秒输出一次评估程序的运行结果。在评估程序多次输入不同的 5.2.1 的样例生成程序生成不同长度的随机字符串，就可以对不同长度的接口切片大小对本地模型遗漏率的性能影响进行评估。

5.2.3 接口算法性能评估实验数据与测试结论

选取 500、1000、1500、2000、2500 切片长度，分别使用模型运行结果评估程序，

各自的切片长度下运行 10 次随机生成字符串的模型遗漏率评估，得到结果评估程序产生的运行结果数据。

该结果评估程序产生的遗漏率评估运行结果数据结果气泡散点图如图 5-13 所示，其中散点图每个点旁边标注的数字含义是：每一次运行产生的 Loss 遗漏关键词次数频次数量。

实验的评估结果和论文前文的第四章 4.1 节切片树方法的理论预测趋势一致：遗漏率 $FN^{[20]}$ （也就是式 5-1 里面的 E_j ）随着输入字符串长度增加，先下降后上升，因此从气泡散点图可以看出，对 OLLama 本地部署的 deepseek-r1:1.5b 模型而言，1000 字符串左右的切片长度遗漏率相对最低。

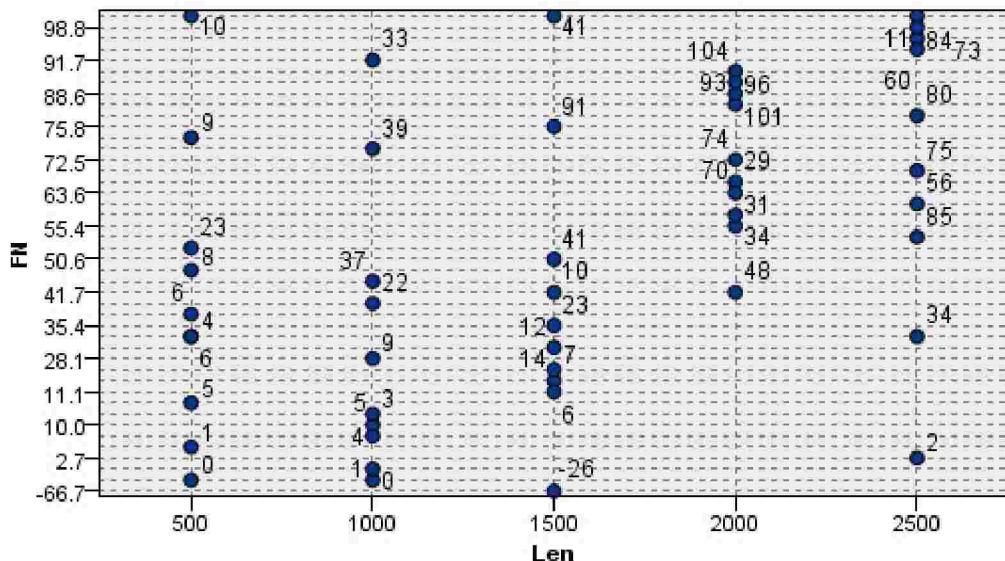


图 5-13 模型运行结果评估程序在样例生成程序生成 50 次结果 Len-FN 散点图

该结果评估程序产生的遗漏率 FN 评估运行结果数据如图 5-14 所示。

根据原始数据计算 500-2500 长度切片的遗漏率 FN 的平均数：

$$\bar{E}_j(L) = \left(\frac{\sum_{o=1}^M (|Loss_o|)}{\sum_{o=1}^M (Tot_o)} \right) \times 100\% \quad \text{式 (5-3)}$$

$$Loss_o = \sum_{i=1}^j (a_i \times n_i) - \sum_{i=1}^j (a_i \times k_i) \quad \text{式 (5-4)}$$

$$Tot_o = \sum_{i=1}^j (a_i \times n_i) \quad \text{式 (5-5)}$$

其中：Loss 是单次评估计算得到的遗漏的关键词数量总数^[28]，Tot 是单次评估计算得到的文本字符串含有关键词总数，M 是同一切片大小进行实验总次数； a_i 是第 i 个关键词的长度， n_i 是第 i 个关键词在文本内实际出现的次数， k_i 是第 i 个关键词在文本内被人工智能模型检测到的输出次数，j 是关键词数量。

得到 500-2500 字符串长度切片处理的平均遗漏率分别为：

$$\bar{E}_j(500) = 72/204 = 35.29\%$$

$$\bar{E}_j(1000) = 153/473 = 32.34\%$$

$$\bar{E}_j(1500) = 271/622 = 43.57\%$$

$$\bar{E}_j(2000) = 680/927 = 73.35\%$$

$$\bar{E}_j(2500) = 560/874 = 64.07\%$$

算出 500-2500 字符串长度切片处理的平均遗漏率在 1000 附近保持较小。

运行的稳定性的遗漏率的标准差 $\sigma(L)$ 是：

$$\sigma(L) = \sqrt{\frac{\sum_{o=1}^M (|E_{j,o}| - \bar{E}_j(L))^2}{M}} \quad \text{式 (5-6)}$$

其中： $E_{j,o}$ 是第 o 次 E_j 实验的遗漏率 FN，j 意思是关键词数量，L 是切片处理窗口长度，M 是同一切片大小进行的实验总次数，从多次实验平均遗漏率数据可知遗漏率 FN 与切片长度 Len 具有明显相关性，切片长度过大时，从 1500 切片长度开始的遗漏率 FN 会显著提高。

得到 500-2500 字符串长度切片处理的遗漏率标准差分别为：

$$\sigma(500) = \sqrt{0.08883233} = 29.72\%$$

$$\sigma(1000) = \sqrt{0.09284333} = 29.78\%$$

$$\sigma(1500) = \sqrt{0.07760021} = 27.86\%$$

$$\sigma(2000) = \sqrt{0.02542005} = 15.94\%$$

$$\sigma(2500) = \sqrt{0.09492233} = 30.81\%$$

由此可见 500-2500 字符串长度切片处理的遗漏率的标准差运行稳定性除了 $\sigma(2000)$ 的实验结果标准差偏小以外，输出稳定性是主要由模型自身逻辑能力决定的，与字符串长度没有明显差异。

以切片长度 Len 为横坐标，遗漏率 FN 为纵坐标，将散点图的标记数字改为处理时间 Time，得到更改了标记数据的实验结果散点图。

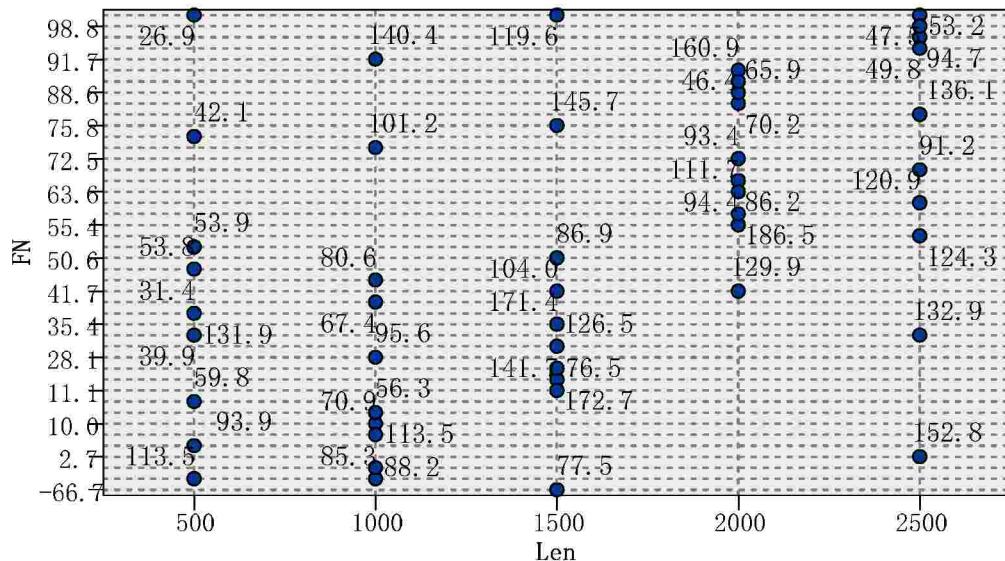


图 5-14 模型运行结果评估程序生成 50 次结果标注有运行时间 Time 的 Len-FN 散点图

设每次运行时间为 t_i ，每次切片长度是 l_i ，那么变量 Time 和切片长度 Len 的皮尔逊相关性（PCC, Pearson correlation coefficient）计算式^[21]是：

$$r_t^l = \frac{\sum_{i=1}^n (t_i - \bar{t})(l_i - \bar{l})}{\sqrt{\sum_{i=1}^n (t_i - \bar{t})^2} \sqrt{\sum_{i=1}^n (l_i - \bar{l})^2}} \quad \text{式 (5-7)}$$

其中 \bar{t} 是平均时间， \bar{l} 是平均长度。

当 len 在 500-1500 范围时，相关系数 $r(500-1500) = 0.6047$ 具有中等相关性。

当 len 长度范围在 500-2000 范围时，相关系数 $r(500-2000) = 0.4257$ 具有弱相关性。

当 len 长度范围在 500-2500 范围时，相关系数 $r(500-2500) = 0.3083$ 具有弱相关性。

在 500-2500 切片长度实验的条件下平均运行时间和运行时间标准差分别为表 5-1：

表 5-1 模型运行结果评估程序生成 50 次运行时间 Time 的平均数、方差和相关性

LEN	500	1000	1500	2000	2500	相关的系数 r
平均数 \bar{t}	64.710	89.940	122.250	104.550	100.340	0.3083
标准差 σ	34.1656	23.1822	34.0478	41.4778	37.1102	

根据散点图 5-14 的纵坐标改为运行时间得到散点图 5-15，图 5-15 横坐标是切片长度 Len，纵坐标是运行时间 Time，散点图点数字标记是遗漏率 FN，这幅散点图表示切片长度 Len 和运行时间 Time 的关系，还表示出运行时间 Time 和同一个实验里遗漏率 FN 的实际实验结果。

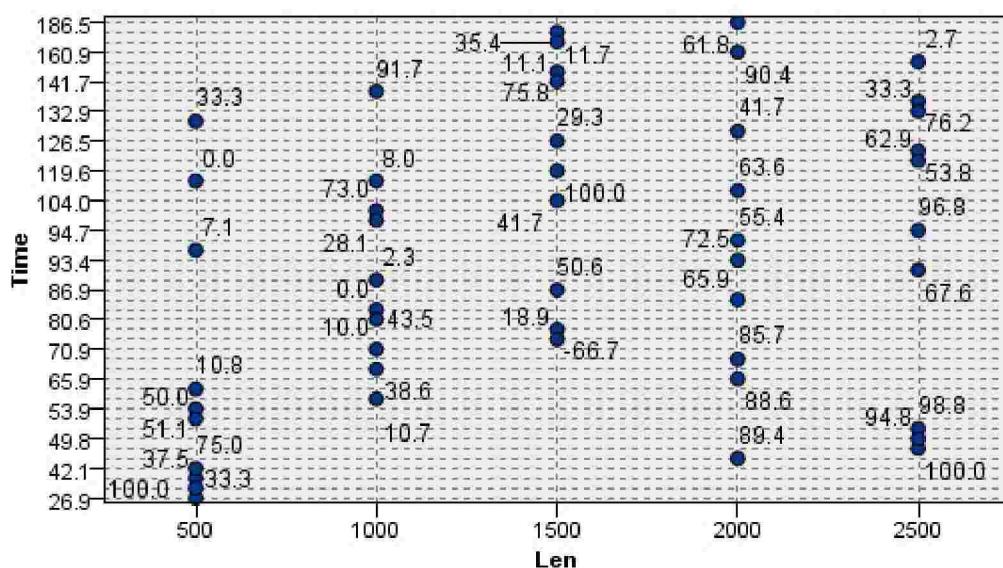


图 5-15 模型运行结果评估程序在样例生成程序生成 50 次结果 Len-Time 散点图

最后，用开发者官方提供的本地 deepseek 工具词库在已有更早的开源工具库 transformer 库的基础上运行 python 语言的 token 词汇单位计算工具 deepseek_tokenizer.py，计算最佳取样切片长度 Len 为 1000 时的 token 词汇数量，与最大 token 词汇单元数量进行比较。

根据本实验结果分析得出如下结论：

1、遗漏率 FN 与切片长度 Len 具有明显的相关性，切片长度 Len 过大时，从 1500 切片长度开始的遗漏率 FN 会显著提高。遗漏率 FN 与切片长度 Len 的相关性与第二章、第四章的切片树理论部分预测切片长度 Len 不宜过大、也不宜过小的理论预测的变化趋势是一致的。切片效果最佳的长度为 1000 字 char 长度，此时 token 词汇数量在 500-800

之间。因此，理论和实验确定的性能最佳的 token 词汇处理状态为 500-800 token 词汇即 1000 字长度的字符串，以实验确定的最佳切片长度 500-800 token 词汇长远小于模型在本地环境下允许运行读入的最大 token 词汇支持数 2048。

2、500-2500 字符串长度切片处理的遗漏率 FN 的标准差运行稳定性除了 $\sigma(2000)$ 的实验结果标准差偏小以外，输出遗漏率 FN 的稳定性是主要由模型自身逻辑能力决定的，与切片长度 Len 没有明显差异。

3、在切片长度范围为 500-1500 时，处理时间 Time 与切片长度 Len 具有中等相关性，当切片长度大于 1500 时，处理时间 Time 与切片长度 Len 的相关性下降。

4、500-2500 字符串长度切片处理的处理时间 Time 的标准差运行稳定性除了 $\sigma(1000)$ 的实验结果标准差偏小以外，处理时间 Time 的稳定性是主要由模型自身逻辑能力决定的，处理时间 Time 的运行稳定性与切片长度 Len 没有明显差异。

5、由散点图 5-15 的散点图纵坐标点数值标记是遗漏率 FN，在切片长度 Len（横坐标）一致时，处理时间 Time 越长（纵坐标越高）时，遗漏率 FN 的数值更有可能越小，模型预测结果更有可能准确。

因此，在本实验最大处理词汇数量为 2048 个词汇 token 的本地运行环境，本实验里的 deepseek-r1:1.5b 本地开源模型均衡适配的切片长度为 1000 个中英文 char 字（不是 token 词汇数量，而是 char 字数）。中英文的切片字数 Len 为 1000 char 字时此时的遗漏率 FN 相对较低、处理时间 Time 运行较为稳定。

5.3 本章的内容小结

本章节介绍了论文所述系统的运行效果和系统的核心算法性能测试评估。

本章第 5.1 节介绍了系统的各个前后端模块的运行软硬件配置和运行效果。本章第 5.1.1 节介绍了系统的软硬件运行环境和运行配置；本章第 5.1.2 节介绍了系统的前端运行效果，本章第 5.1.3 节介绍了系统的后端运行效果。

本章第 5.2 节介绍了系统的核心算法性能测试评估，其中第 5.2.1 节介绍了输入测试样例生成程序，其中第 5.2.2 节介绍了模型的运行结果评估程序，5.2.3 节最终总结了第 4 章 4.1 节介绍的切片树 Slice-Tree 接口算法切片性能评估实验数据与测试结论。

第六章 总结

本课题设计并实现了一个 PDF 的开发一个接入 DeepSeek-R1-1.5B 本地部署的开源平台 Ollama 模型接口的 PDF 机器学习摘要图表理解系统，基于指令提示工程实现基于大模型的图和表的理解。

本课题设计一套基本交互界面，在指定文件夹导入 PDF 并基于界面实现文字内容和表格内容的问答。能够根据文字，生成易于理解的摘要。进行系统测试和性能评估，确保系统的可用性。

如果用户实现了登陆，则支持根据用户选择存储用户登陆以后生成摘要的历史记录。论文的整体内容包括系统设计、实现、测试及分析等内容，软件的数据库使用基于 node.js V14 的 vue 2 在 Visual Studio 编辑整合 MongoDB 数据库。

本课题主要工作是进行机器学习 PDF 文档摘要生成器服务平台与接口的后端开发。在理解机器学习原理知识和 PDF 文档接口处理知识的基础上，分析项目需求，根据项目需求进行系统设计和功能设计，采用 Ollama 与 Visual Studio 编辑和 node.js 开源进程间通信编译器部署的 Vue 2 项目框架联合开发，实现系统设计与功能设计，满足用户需求。

由于大模型直接读取字符串长度受到模型本身尺寸的限制和运算性能的限制，因此本课题研究的 SliceTree 切片读取长文本延长处理时间的方式生成摘要可以使用减少单次运算长度、延长处理时间的方式在性能相对低于服务器的低端机器上部署和运行 Deepseek 相对小体量版本的模型、进行相对更加廉价容易部署的长文本处理。

本课题的研究不只是适用于 PDF 文件，而且还为相对廉价机器进行本地化的长文本处理提供了一种可行的解决思路。

参考文献

- [1] DeepSeek-AI, Daya Guo, Dejian Yang. DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning[R/OL]. <https://arxiv.org/abs/2501.12948>.
- [2] DeepSeek-AI, Xiao Bi, Deli Chen. DeepSeek LLM: Scaling Open-Source Language Models with Longtermism [R/OL]. <https://arxiv.org/abs/2401.02954>.
- [3] DeepSeek-AI, Aixin Liu, Bei Feng. DeepSeek-V3 Technical Report [R/OL]. <https://arxiv.org/abs/2412.19437>.
- [4] Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri. The Llama 3 Herd of Models[R/OL]. <https://arxiv.org/abs/2407.21783v3>.
- [5] Yilun Zhao, Lyuhao Chen, Arman Cohan, and Chen Zhao. TaPERA: Enhancing Faithfulness and Interpretability in Long-Form Table QA by Content Planning and Execution-based Reasoning[C]// Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics, Bangkok, Thailand: Association for Computational Linguistics, 2024, 1: 12824-12840.
- [6] The International Organization for Standardization. Document management — Portable document format — Part 1: PDF 1.7[S].
- [7] Yan Song, Shuming Shi, Jing Li, and Haisong Zhang. Directional Skip-Gram: Explicitly Distinguishing Left and Right Context for Word Embeddings[C]// Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, New Orleans, Louisiana: Association for Computational Linguistics, 2018: 175-180.
- [8] Chujie Zheng, Minlie Huang, and Aixin Sun. 2019. ChID: A Large-scale Chinese IDiom Dataset for Cloze Test[C]// Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, Florence, Italy: Association for Computational Linguistics, 2019: 778-787.
- [9] Biao Zhang, Rico Sennrich. Root Mean Square Layer Normalization[C]// Proceedings of the 33rd International Conference on Neural Information Processing Systems, New York, United States: Curran Associates Inc, 2019: 12381-12382.
- [10] N. Shazeer. Glu variants improve transformer[R\OL]. <https://arxiv.org/abs/2002.05202>.
- [11] 葛旭冉, 欧洋, 王博, 赵宇, 吴利舟, 王子聪, 陈志广, 肖依. 大语言模型推理中的存储优化技术综述[J]. 计算机研究与发展, 2025, 62(3): 545-562.
- [12] 尚碧筠, 韩银俊, 肖蓉, 陈正华, 屠要峰, 董振江. ScaleFS: 面向大语言模型的高性能可扩展元

数据设计[J]. 计算机研究与发展, 2025, 62(3): 589-604.

[13] Prajit Ramachandran, Barret Zoph, and Quoc VLe. Searching for activation functions[R\OL].
<https://arxiv.org/abs/1710.05941>.

[14] J. Ainslie, J. Lee-Thorp, M. de Jong, Y. Zemlyanskiy, F. Lebrón, and S. Sanghai. Gqa: Training generalized multi-query transformer models from multi-head checkpoints[R\OL].
<https://arxiv.org/abs/2305.13245>.

[15] Aran Komatsuzaki, Joan Puigcerver, James Lee-Thorp, Carlos Riquelme Ruiz, Basil Mustafa, Joshua Ainslie. Sparse upcycling: Training mixture-of-experts from dense checkpoints[R\OL].
<https://arxiv.org/abs/2212.05055>.

[16] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory[J]. Neural computation, 1997, 9(8):1735–1780.

[17] Karl Moritz Hermann, Tomáš Kočiský, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, Phil Blunsom. Teaching machines to read and comprehend[C]// NIPS'15: Proceedings of the 29th International Conference on Neural Information Processing Systems - Volume 1, 55 Hayward St.Cambridge MA United States: MIT Press, 2015: 1693-1701.

[18] Wei An, Xiao Bi, Guanting Chen, Shanhua Chen, Chengqi Deng, Honghui Ding. Sparse upcycling: Fire-Flyer AI-HPC: A Cost-Effective Software-Hardware Co-Design for Deep Learning [C]// SC '24: Proceedings of the International Conference for High Performance Computing, Networking, Storage, and Analysis, 2024, Article 83: 1 – 23.

[19] Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, Yunfeng Liu. RoFormer: Enhanced transformer with Rotary Position Embedding[J]. Neurocomputing, 2024, 568.

[20] Wu W, Yu Z, He J. A Semi-Supervised Deep Network Embedding Approach Based on the Neighborhood Structure[J]. Big Data Mining and Analytics, 2019, 2(3): 205-216.

[21] Mohy-Eddine M, Guezzaz A, Benkirane S, et al. An Ensemble Learning Based Intrusion Detection Model for Industrial IoT Security[J]. Big Data Mining and Analytics, 2023, 6(3): 273-287.

[22] Malek YN, Najib M, Bakhouya M, et al. Multivariate Deep Learning Approach for Electric Vehicle Speed Forecasting[J]. Big Data Mining and Analytics, 2021, 4(1): 56-64.

[23] Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. Self-attention with relative position representations[C]// The 2018 Conference of the North American Chapter of the Association for Computational Linguistics: HumanLanguage Technologies, 2018.

- [24] Zihang Dai, Z. Yang, Yiming Yang, J. Carbonell, Quoc V. Le, and R. Salakhutdinov. Transformer-xl: Attentive language models beyond a fixed-length context[C]// ACL 2019: 57th Annual Meeting of the Association for Computational Linguistics, 2019.
- [25] 毕鑫,聂豪杰,赵相国,袁野,王国仁.面向知识图谱约束问答的强化学习推理技术[J].软件学报,2023,34(10):4565-4583.
- [26] J. Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding[C]// The 2019 Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, 2019.
- [27] Chin-Yew Lin and Eduard Hovy. Automatic evaluation of summaries using n-gram co-occurrence statistics[C]// Proceedings of the 2003 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics, 2003: 150–157.
- [28] Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. 2020. Bertscore: Evaluating text generation with bert[C]// International Conference on Learning Representations, 2020.
- [29] Danqi Chen, Jason Bolton, and Christopher D Manning. A thorough examination of the cnn/daily mail reading comprehension task[C]// Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), 2016: 2358–2367.
- [30] Anqi Mao, Mehryar Mohri, Yutao Zhong. Cross-entropy loss functions: theoretical analysis and applications[C]// ICML'23: Proceedings of the 40th International Conference on Machine Learning, 2023: 23803-23828.