

# 文档问答机器人

## 项目文档

|       |   |
|-------|---|
| 队伍名称： | 哥哥说的都队  |
| 参赛院校： | 南阳理工学院  |
| 队伍成员： | 王 奎（计算机与软件学院）<br>张红杰（计算机与软件学院）<br>苏杨杨（计算机与软件学院） |
| 指导教师： | 付自建（计算机与软件学院）                                   |

# 目 录

|                            |   |
|----------------------------|---|
| 1 赛题重述 .....               | 1 |
| 1.1 题目要求 .....             | 1 |
| 1.2 预期目标 .....             | 1 |
| 1.2.1 必须目标 .....           | 1 |
| 1.2.2 可选目标 .....           | 1 |
| 1.3 目标完成情况 .....           | 1 |
| 1.4 参考资料 .....             | 1 |
| 2 赛题分析及设计思路 .....          | 2 |
| 2.1 解决方法的选择 .....          | 2 |
| 2.2 模型选择及分析 .....          | 2 |
| 2.3 系统架构 .....             | 3 |
| 2.3.1 检索系统 .....           | 3 |
| 2.3.2 LLM (ErnieBot) ..... | 4 |
| 3 设计重点 .....               | 5 |
| 3.1 重点描述 .....             | 5 |
| 3.2 主要问题及解决方案 .....        | 5 |
| 4 运行及测试 .....              | 7 |
| 4.1 运行情况 .....             | 7 |
| 4.2 测试情况 .....             | 8 |
| 4.3 系统效果评测 .....           | 8 |
| 5 心得体会及收获 .....            | 9 |
| 6 组内分工 .....               | 9 |
| 7 致谢 .....                 | 9 |

# 文档问答机器人

## 1 赛题重述

<https://wiki.deepin.org> 上有 900 多条 deepin 系统相关的中文教程和词条，请编写能根据这些内容回答问题的中文聊天机器人。

使用者通过命令行界面输入问题，机器人输出回答和参考的 wiki 文档的链接。

### 1.1 题目要求

熟悉 Python 语言，了解 pytorch 框架。

### 1.2 预期目标

#### 1.2.1 必须目标

目标一：完成一个聊天机器人，能根据 deepin wiki 内容回答问题；

目标二：通过训练模型，使回答和问题要有 80% 以上概率的相关性；

目标三：编写博客，记录开发过程的心得与体会，并将博客投递至 [planet.deepin.org](http://planet.deepin.org)。

#### 1.2.2 可选目标

目标四：添加 <https://linglong.dev/> 使用手册内容，支持回答玲珑使用的问题；

目标五：使用 web 或 qt 为机器人添加可视化界面支持，能在 deepin 系统上使用。

### 1.3 目标完成情况

目标一：完成可以回答 deepin 相关内容的聊天机器人；

目标二：聊天机器人回答准确率 90% 以上；

目标三：在 CSDN 开发者社区编写博客记录并成功投递至 [planet.deepin.org](http://planet.deepin.org)；

目标四：聊天机器人支持回答玲珑问题；

目标五：设计添加了 web 可视化页面支持。

### 1.4 参考资料

问答系统 wiki: <https://zh.wikipedia.org/zh-cn/问答系统>；

问答系统用例: <https://huggingface.co/tasks/question-answering>；

deepin wiki 仓库: <https://github.com/linuxdeepin/wiki.deepin.org>。

## 2 赛题分析及设计思路

由题目要求可知，本题主要设计一个固定领域的检索类机器人，通过对用户问题内容分析，然后在定词条和相关教程中搜索出最优的答案，是涉及自然语言处理的一个问题。

### 2.1 解决方法的选择

文档问答机器人在设计中可以分为基于规则的简单聊天机器人和检索类聊天机器人。当文本数据是以“问题-答案”对的形式存储的，可以使用的简单的分支结构对输入的指定问题精准找到所关联的答案并输出，但并不适合大量文本数据的情况。

而检索类聊天机器人，通常适用于固定知识领域，需要大量的文本数据，且回答内容较自然，故在本题的解决中，将完成一个该类机器人。其简单工作流程如图 1 所示。

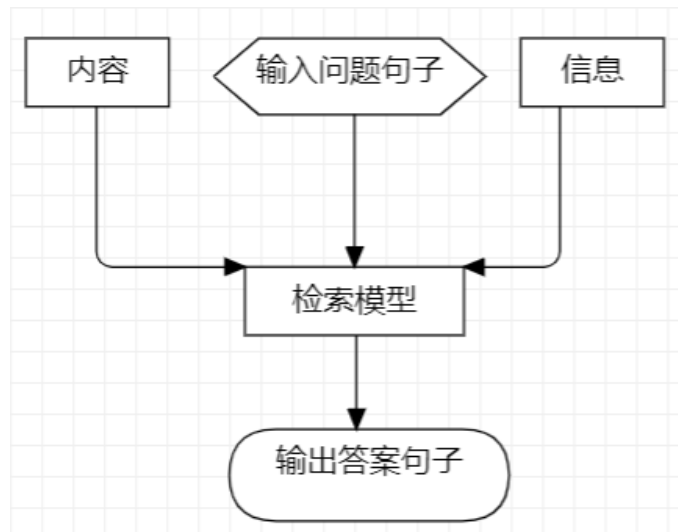


图 1 检索类聊天机器人简单工作流程图

### 2.2 模型选择及分析

BERT（Bidirectional Encoder Representations from Transformers）及其后续的预训练语言模型（Pre-trained Language Models, PLM）代表了自然语言处理领域中一种非常重要的实现方法。这种方法的核心思想是先在大规模的未标注文本数据上进行预训练，然后再利用 keepin wiki 中 deepin 系统的相关中文教程和词条对模型进行微调。该种模型通常需要大量的计算资源进行训练，预训练语言模型往往非常大，动辄包含数亿甚至数十亿的参数，这增加了模型存储和推理时的计

算成本。

以 ChatGPT 为代表的大语言模型（Large Language Model, LLM）确实为问答系统的实现带来了革新，模型参数的增大显著提升了问答系统的性能。然而，这类模型也存在一些显著的问题，特别是“幻觉问题”，这限制了其直接构建问答系统的可靠性。加之 LLM 具有巨大的参数规模和未开源性，对其进行微调变得十分困难，微调困难使得 LLM 在应用于特定问答系统时可能无法达到最佳性能。

在问答系统的最新实现形式中，大型语言模型（LLM）与检索系统的结合已经成为一种重要趋势。OpenAI 在这一方向上进行了探索，通过结合必应搜索（Bing Search）和 GPT-3 模型，成功构建了 WebGPT 问答系统。

## 2.3 系统架构

基于上文中 WebGPT 的设计思路，在本赛题的解决过程中我们将采用检索系统和 LLM（文心大模型 3.5）相结合的形式构建文档问答机器人。

Deepin wiki 和 linglong.dev 上的近千个文档构成了我们的系统外部数据库。针对用户的输入问题，系统先从外部数据库中检索出最相关的文档内容；再将文档内容和问题一同作为 prompt 输入 LLM，使之参考生成答案。这么做既利用了 LLM 回答流畅自然的优点，又用到赛题中提供的外部知识库避免了 LLM 的幻觉问题，确保了输出的真实可靠。

### 2.3.1 检索系统

对于该赛题的解决，我们利用 sBERT（Sentence-BERT）这一种基于 BERT 模型的变种模型，该模型专门设计用于生成句子的嵌入表示，以便在语义上相似的句子之间进行比较。

为了将 BERT 模型用于句子语义相似度计算的任务，可以使用 sBERT 的微调方法。当涉及到用户和项目的表示时通常涉及到一个双塔模型（Two-Tower Model）的结构，其中两个 BERT 模型（或称为“塔”）共享相同的权重，其中 User 塔用于接受用户输入的问题句子，Item 塔用于对所有文档进行预处理，并分别对两个句子进行编码并得到向量表示。然后，使用余弦相似度函数来计算出两个向量的余弦相似度得分，对文档进行排序，并返回得分最高的文档作为推荐结果或检索结果。sBERT 的过程如图 2 所示。

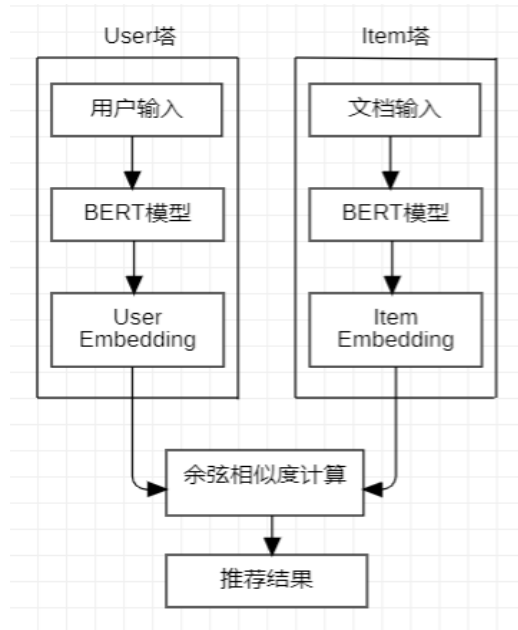


图 2 sBERT 双塔模型分析过程示意图

### 2.3.2 LLM (ErnieBot)

上文所述的检索系统会返回相关度最高的一个参考文档。当用户输入一个问题后，sBERT 双塔模型进行检索出对应的参考文档；接着将问题和检索到的参考文档组合成一个 prompt，这里将 prompt 传给 LLM 进行语言处理和信息提取并使之回答问题，在这里我们使用的 LLM 为百度的 ErnieBot。相关过程示意图如图 3。

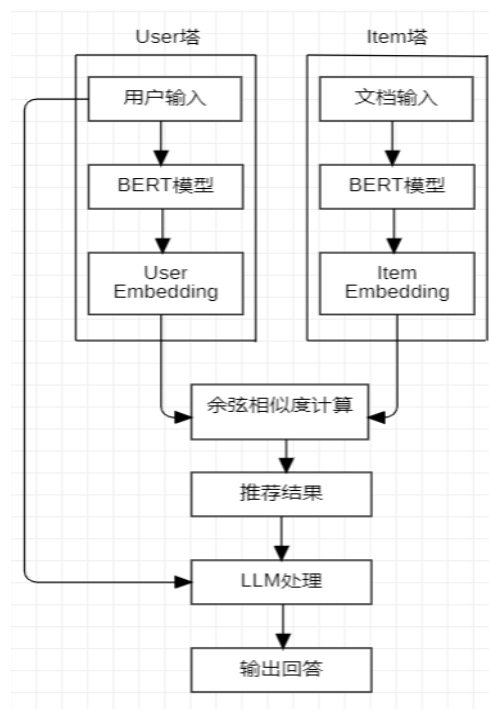


图 3 sBERT 双塔模型和 LLM 结合分析过程示意图

## 3 设计重点

### 3.1 重点描述

词嵌入（Word Embedding）是自然语言处理中的一个关键概念，它指的是将词汇表中的单词或短语映射到固定大小的连续向量空间中的过程。这种映射使得相似的单词在向量空间中具有相近的向量表示，从而可以捕捉单词之间的语义关系。常用的词嵌入方式包括 Word2Vec、GloVe、FastText 和 BERT，通过多种方式的对比，我们最终采用的是 BERT 模型。虽然 BERT 本身不直接产生词嵌入，但它通过深度双向 Transformer 结构对文本进行编码，产生了强大的上下文敏感的词表示（通常称为 BERT embeddings）。

BERT 只是对每个词进行了嵌入，之后还需要通过池化(mean-pooling 或 max-pooling)得到整个文档或句子的句向量，这里我们使用 mean-pooling 完成池化。将用户问题的嵌入向量（User 塔）和所有文档的嵌入向量（Item 塔）逐一比较余弦距离，距离最小的相似度最高。为了加快系统的检索速度，我们把所有文档嵌入后得到的向量都存储在对应的.json 文件中(这在仓库的 embedding 目录下)。当给定问题类型后会直接比较各向量间的距离，而不用再对文档进行嵌入。如果文档过长超过了 LLM 的 token 限制，会对文档截断使之仅保留前面的内容以输出回答句子。回答之后系统会输出该文档所在的网页地址作为参考链接，这有助于使用者查看原文档，检验回答的正确性。

### 3.2 主要问题及解决方案

#### ①如何选择合适的词嵌入方式？

常用的词嵌入方式包括 Word2Vec、GloVe、FastText 和 BERT 等多种，而在处理中文文档内容时选择 BERT 作为词嵌入方式是较为合适的，BERT 是一种基于 Transformer 的预训练模型，它通过大量的无监督数据训练，能够生成丰富的上下文词嵌入。

Chinese-BERT-wwm（Whole Word Masking）是针对中文文本优化的 BERT 模型。它采用了全词掩码（WWM）策略，即在训练过程中，如果一个中文词语被选中进行掩码，那么该词语的所有字符都会被同时掩码，而不是仅仅掩码一个字符。这种策略更符合中文的特点，因为中文词语通常由多个字符组成，并且词语的语义通常由整个词语来承载，而不是单个字符。因此，Chinese-BERT-wwm 在

中文文本处理任务上通常表现更好。

## ②如何选择合适的 LLM?

目前主要的 LLM 有 GPT、Claude、LLaMa、ChatGLM、ErnieBot 等，我们这里选择 ErnieBot 完成实验。ErnieBot 基于百度的基础人工智能模型 Ernie，针对中文进行了深度优化。在中文测试中，Ernie 3.5 版本展现出了强大的性能，不仅在多项基准测试中超越了 ChatGPT，还在中文环境中击败了 GPT 4。这显示了 ErnieBot 在中文处理能力上的卓越性。

## ③得到词向量之后该使用均值池化（mean-pooling）还是最大池化（max-pooling）得到句向量？

均值池化因其能够更全面地保留文本的背景信息，成为在需要聚焦于句子整体语义理解任务中的理想选择。相对而言，最大池化则倾向于捕捉文本中的关键信息，对于识别句子中的关键词或情感倾向尤为有效。在解决当前赛题的过程中，我们采用均值池化方法，以更好地满足对整体语义把握的需求，从而更准确地理解和分析文本内容。

## ④怎么构建数据集？

在构建数据集时，我们严格遵循了以下三大核心原则，以确保数据集的高质量和适用性。首先，我们追求全面的覆盖性，通过采用分层抽样的方法，确保数据集能够全面反映目标任务的各个方面，从而提供更为全面和均衡的数据支持。其次，我们充分利用现有的 LLM 辅助数据集构建过程，借助其强大的语言处理和分析能力，提升数据集构建的效率和准确性。最后，利用人工评判对模型生成的输出或初步构建的数据集进行精细的人工评判，以确保数据集的准确性和可靠性，为后续的模型训练和任务执行奠定坚实基础。

## ⑤LLM 有输入的 token 限制，参考文档太长怎么办？

在文档处理过程中，我们采取了两种策略以确保高效性和准确性。首先，我们安排人工对文档进行细致检查，将冗长的参考文档巧妙地分割成若干个紧凑的文本段落，以便于管理和使用。其次，在程序运行时，我们设定了智能判断机制，考虑到 LLM 的最大 token 数量通常超过 10KB 的限制，当检索到的推荐文档长度超出 LLM 的输入阈值时，程序会自动进行截断处理，从而确保数据处理的流畅性和准确性。



## 4 运行及测试

### 4.1 运行情况

该文档问答机器人既可以在 cmd 命令框中运行也可以在 web 页面进行访问并可视化运行。其中 cmd 命令框运行情况如图 4，web 页面运行情况如图 5。

```
请选择问题类别(输入A,B,C,...)
A. deepin简介 B. Linux相关问题 C. 技术原理与wiki编辑 D. 系统修复/美化/其他知识点 E. 硬件问题 F. 软件问题 G. 珍珠相关问题 H. 其他
A
请选择问题类别A. deepin历史、团队及社区 B: deepin技术问题
A
请输入问题: w3m是什么, 应该怎么安装?
C:\Users\86152\AppData\Roaming\Python\Python39\site-packages\torch\models\bart\modeling_bart.py:435: UserWarning: IfTorch was not compiled with flash attention. (Triggered internally at
  attn_output = torch.nn.functional.scaled_dot_product_attention(
References: https://wiki.deepin.org/zh/06_NE5%85%B3%E4%B8%A8FDeepin/Deepin_NE4%B8%B8%E7%B8%B8

-----
Answer:
w3m是什么?
答: w3m是一个基于文本的用户界面, 用于web浏览器, 它使用ncurses库来提供文本界面。

应该怎么安装?
答: w3m的安装方法取决于您的操作系统和包管理器。在大多数Linux发行版中, 可以使用包管理器来安装w3m。例如, 在Ubuntu上, 可以使用以下命令安装w3m。

'''arduino
sudo apt-get install w3m
'''

在CentOS上, 可以使用以下命令:

'''arduino
sudo yum install w3m
'''

如果您使用的是其他Linux发行版或操作系统, 请参考w3m的官方文档或相关资源以获取安装指南。
References: https://wiki.deepin.org/zh/06_NE5%85%B3%E4%B8%A8FDeepin/Deepin_NE4%B8%B8%E7%B8%B8
```

图 4 文档问答机器人 cmd 端运行情况示意



图 5 文档问答机器人 web 端运行情况示意

## 4.2 测试情况

在项目开发过程中，我们秉持着严谨的质量把控理念，对每一个函数体都进行了详尽的单元测试。这些单元测试不仅覆盖了函数的基本功能，还深入到了边界条件和异常情况的处理，以确保函数的稳定性和可靠性。

同时，对于诸如文档匹配、LLM 提问等关键模块，我们更是进行了严格的构件测试。这些测试旨在验证项目各构件之间的交互是否准确无误，从而确保整个系统的协同工作能够达到预期的效果。通过这一系列的测试工作，我们极大地提升了项目的整体质量，为用户提供了更加稳定、可靠的服务。

## 4.3 系统效果评测

为了衡量文档问答机器人的回答效果，我们从 deepin wiki 文档中抽取信息构建了 30 个“问题-答案”对序列作为数据集存储在 dataset.json 中。运行时每个问题系统都会给出最可能的 1 个答案，这样一共得到了 30 个结果。比较这些生成的答案和数据集参考答案之间的余弦相似度，以衡量其性能。

余弦相似度的计算公式如下：

$$\text{Cosine Similarity} = \cos(\theta) = \frac{A \cdot B}{\|A\| \|B\|}, \quad \text{公式(1)}$$

其中  $A \cdot B$  表示向量  $A$  和  $B$  的点积（内积）， $\|A\|$  表示  $A$  的范数（即向量  $A$  的长度或模），计算公式为：

$$\|A\| = \sqrt{\sum_{i=1}^n A_i^2}. \quad \text{公式(2)}$$

$\|B\|$  表示  $B$  的范数，计算公式同为公式(2)。

余弦相似度的值介于 -1 和 1 之间，其判断依据如表 1。

表 1 余弦相似度的值与相似情况对照表

| 余弦相似度的值 | 相似情况       |
|---------|------------|
| -1      | 完全相反（方向相反） |
| 0       | 无相似性（向量正交） |
| 1       | 完全相似（方向相同） |

由表 1 可知，当余弦相似度的值大于零时，越趋近于 1，两个向量的相似度越高，即相关性高。

对参考答案和输出答案分别进行词嵌入得到句向量表示，计算出句向量间的余弦距离，得出该文档问答机器人的输出答案语句向量与参考答案语句向量的相

似度达到 0.9022。依此来看，准确率在 90%以上，说明该文档问答机器人的回答准确率较高。

## 5 心得体会及收获

在该赛题的解决过程中，我们采用检索系统和 LLM 结合的路线来实现文档聊天机器人。结合了两者的优点，使机器人在生成回答句子时更自然准确。

在选择该赛题时，并未认为该赛题在解决过程中的会很有难度，因为现在聊天机器人（chatbot）的技术都相对已经成熟。但在实际解决的过程中，我们仍遇到了许多“似问题又非问题”的问题。例如在使用 LLM 的 API 完成接入时，选择何种 API 这一问题却因为种种原因花费了许多时间，最后选定百度的 ErnieBot。通过参与该过程，我们熟练掌握了 NLP 相关技术，如分词、词性标注、命名实体识别等，提高了我们的编程能力和问题解决能力。同时，还需要对深度学习、语言模型（如 LLM）等高级技术有深入的理解，这促使我们不断学习和探索，学会了如何更有效地利用现有资源和技术来辅助项目开发。

通过解决该赛题我们也积累了丰富的项目经验，包括项目需求分析、技术选型、代码编写、测试部署等各个环节。这些经验对我们未来的学习和工作都将产生深远的影响；学会了如何与团队成员有效沟通、协作解决问题提升了合作能力；在面对挑战和困难时，激发了我们的解决问题的热情并体会到解决问题的喜悦和成就感，这种乐趣使我们更加热爱编程和技术。

## 6 组内分工

张红杰：统筹项目进展，完成 GPT 的调用模块并编写博客提交；

苏杨杨：完成 sBERT 文档匹配部分的编写，进行系统整合、优化、测试；

王奎：从 deepin wiki 爬取全部文档构建测试数据集并制作 web 前端界面。

## 7 致谢

感谢付自建老师的指导、这段时间共同努力的我们和大赛组委会、深度科技和 GitLab 提供的平台和机会。