

基于LLM+向量库的文档对话经验面

1. 基于LLM+向量库的文档对话 基础面

1. 💡 LLMs 存在模型幻觉问题，请问如何处理？

大语言模型的模型幻觉问题是指其可能生成看似合理但实际上不准确或不符合事实的内容。这是由于大语言模型在训练过程中接触到的数据源的偏差、噪声或错误所导致的。处理大语言模型的模型幻觉问题需要采取一些方法和策略，以下是一些建议：

1. 数据清洗和预处理：在训练大语言模型之前，对数据进行仔细的清洗和预处理是至关重要的。删除不准确、噪声或有偏差的数据可以减少模型幻觉问题的出现。
2. 多样化训练数据：为了减少模型对特定数据源的依赖和偏好，可以尽量使用多样化的训练数据。包括来自不同领域、不同来源和不同观点的数据，以获得更全面的语言理解。
3. 引入多样性的生成策略：在生成文本时，可以采用多样性的生成策略来减少模型的倾向性和幻觉问题。例如，使用温度参数来调整生成的多样性，或者使用抽样和束搜索等不同的生成方法。
4. 人工审核和后处理：对生成的文本进行人工审核和后处理是一种常用的方法。通过人工的干预和修正，可以纠正模型幻觉问题，并确保生成的内容准确和可靠。
5. 引入外部知识和约束：为了提高生成文本的准确性，可以引入外部知识和约束。例如，结合知识图谱、实体识别或逻辑推理等技术，将先验知识和约束融入到生成过程中。

这些方法可以帮助减少大语言模型的模型幻觉问题，但并不能完全消除。因此，在使用大语言模型时，仍然需要谨慎评估生成结果的准确性和可靠性，并结合人工的审核和后处理来确保生成内容的质量。

2. 💡 基于LLM+向量库的文档对话 思路是怎么样？

基于大语言模型和向量库的文档对话可以通过以下实现思路：

1. 数据预处理：首先，需要对文档数据进行预处理。这包括分词、去除停用词、词干化等步骤，以准备文档数据用于后续的向量化和建模。
2. 文档向量化：使用向量库的方法，将每个文档表示为一个向量。常见的向量化方法包括TF-IDF、Word2Vec、Doc2Vec等。这些方法可以将文档转换为数值向量，以便计算文档之间的相似度或进行聚类分析。
3. 大语言模型训练：使用大语言模型，如GPT、BERT等，对文档数据进行训练。这样可以使模型学习到文档之间的语义关系和上下文信息。
4. 文档检索：当用户提供一个查询文本时，首先对查询文本进行向量化，然后计算查询向量与文档向量之间的相似度。可以使用余弦相似度或其他相似度度量方法来衡量它们之间的相似程度。根据相似度排序，返回与查询文本最相关的文档。
5. 文档推荐：除了简单的文档检索，还可以使用大语言模型生成推荐文档。通过输入用户的查询文本，使用大语言模型生成与查询相关的文本片段或摘要，并根据这些生成的文本片段推荐相关的文档。
6. 对话交互：在文档对话系统中，用户可以提供多个查询文本，并根据系统的回复进行进一步的对话交互。可以使用大语言模型生成系统的回复，并根据用户的反馈进行迭代和改进。

通过以上实现思路，可以构建一个基于大语言模型和向量库的文档对话系统，使用户能够方便地进行文档检索、推荐和对话交互。具体的实现细节和技术选择会根据具体的应用场景和需求来确定。

3. 💡 基于LLM+向量库的文档对话 核心技术是什么？

基于大语言模型和向量库的文档对话的核心技术包括以下几个方面：

1. 大语言模型：大语言模型是指能够理解和生成人类语言的深度学习模型，如GPT、BERT等。这些模型通过在大规模文本数据上进行预训练，学习到语言的语义和上下文信息。在文档对话系统中，大语言模型可以用于生成回复、推荐相关文档等任务。
2. 文档向量化：文档向量化是将文档表示为数值向量的过程。这可以使用向量库技术，如TF-IDF、Word2Vec、Doc2Vec等。文档向量化的目的是将文档转换为计算机可以处理的数值形式，以便计算文档之间的相似度或进行其他文本分析任务。
3. 相似度计算：相似度计算是文档对话系统中的重要技术。通过计算查询文本向量与文档向量之间的相似度，可以实现文档的检索和推荐。常见的相似度计算方法包括余弦相似度、欧氏距离等。
4. 对话生成：对话生成是指根据用户的查询文本生成系统的回复或推荐文档。这可以使用大语言模型来生成自然语言的回复。生成的回复可以基于查询文本的语义和上下文信息，以提供准确和有意义的回复。
5. 对话交互：对话交互是指用户和系统之间的交互过程。用户可以提供查询文本，系统根据查询文本生成回复，用户再根据回复提供进一步的查询或反馈。对话交互可以通过迭代和反馈来改进系统的回复和推荐。

这些技术共同构成了基于大语言模型和向量库的文档对话系统的核心。通过结合这些技术，可以实现文档的检索、推荐和对话交互，提供更智能和个性化的文档服务。

4. 💡 基于LLM+向量库的文档对话 prompt 模板 如何构建？

构建基于大语言模型和向量库的文档对话的prompt模板可以考虑以下几个方面：

1. 查询类型：首先确定用户可能的查询类型，例如问题查询、主题查询、摘要查询等。针对不同的查询类型，可以构建相应的prompt模板。例如，对于问题查询，可以使用"我有一个关于XXX的问题"作为模板；对于主题查询，可以使用"我想了解关于XXX的信息"作为模板。
2. 查询内容：根据文档的特点和领域知识，确定用户可能会查询的内容。例如，对于新闻文档，查询内容可以包括新闻标题、关键词、时间范围等；对于学术论文，查询内容可以包括作者、论文标题、摘要等。根据查询内容，可以构建相应的prompt模板。例如，对于查询新闻标题的情况，可以使用"请问有关于XXX的新闻吗？"作为模板。
3. 上下文信息：考虑上下文信息对于查询的影响。用户之前的查询或系统的回复可能会影响当前的查询。可以将上下文信息加入到prompt模板中，以便更好地理解用户的意图。例如，对于上一轮的回复是关于某个主题的，可以使用"我还有关于上次谈到的XXX的问题"作为模板。
4. 可变参数：考虑到用户的查询可能有不同的变化，可以在prompt模板中留出一些可变的参数，以便根据具体查询进行替换。例如，可以使用"我想了解关于XXX的信息"作为模板，其中的XXX可以根据用户的查询进行替换。

通过这些方面的考虑，可以构建多个不同的prompt模板，以满足不同类型和内容的查询需求。在实际应用中，可以根据具体的场景和数据进行调整和优化，以提供更准确和有针对性的查询模板。

2. 基于LLM+向量库的文档对话 优化面

1. 💡 痛点1：文档切分粒度不好把控，既担心噪声太多又担心语义信息丢失

在基于大语言模型和向量库的文档对话中，确实需要在文档切分的粒度上进行权衡。如果切分得太细，可能会引入较多的噪声；如果切分得太粗，可能会丢失一些重要的语义信息。以下是一些解决方案：

1. 预处理和过滤：在进行文档切分之前，可以进行一些预处理和过滤操作，以减少噪声的影响。例如，可以去除文档中的停用词、标点符号、特殊字符等，以及进行拼写纠错和词形还原等操作。这样可以降低噪声的存在，提高文档切分的质量。
2. 主题建模：可以使用主题建模技术，如LDA (Latent Dirichlet Allocation) 等，对文档进行主题抽取。通过识别文档的主题，可以帮助确定文档切分的粒度。例如，将同一主题下的文档划分为一个切分单元，以保留更多的语义信息。
3. 上下文信息：在进行文档切分时，考虑上下文信息对于语义的影响。例如，将与上一文档相关联的文档划分为一个切分单元，以保留上下文的连贯性和语义关联。这样可以更好地捕捉文档之间的语义信息。
4. 动态切分：可以采用动态切分的方式，根据用户的查询和需要，实时生成切分单元。例如，根据用户的关键词或查询意图，动态生成包含相关信息的切分单元，以减少噪声和提高语义的准确性。
5. 实验和优化：在实际应用中，可以进行一系列的实验和优化，通过不断调整和评估文档切分的效果。可以尝试不同的切分粒度，评估其噪声和语义信息的平衡。通过实验和优化，逐步找到合适的文档切分策略。

综上所述，解决文档切分粒度的问题需要综合考虑预处理、主题建模、上下文信息、动态切分等多个因素，并通过实验和优化来找到最佳的平衡点，以保留足够的语义信息同时减少噪声的影响。

2. 💡 痛点2：在基于垂直领域 表现不佳

如果在垂直领域中，基于LLM (Language Model + Retrieval) 和向量库的文档对话表现不佳，可以考虑以下方法来改进：

1. 针对垂直领域进行领域特定训练：LLM模型是基于大规模通用语料库进行训练的，可能无法充分捕捉垂直领域的特点和术语。可以使用领域特定的语料库对LLM模型进行微调或重新训练，以提高在垂直领域的表现。
2. 增加领域知识：在向量库中，可以添加垂直领域的专业知识，如领域术语、实体名词等。这样可以提高向量库中文档的表示能力，使其更适应垂直领域的对话需求。
3. 优化检索算法：在使用向量库进行文档检索时，可以尝试不同的检索算法和相似度计算方法。常用的算法包括余弦相似度、BM25等。通过调整参数和算法选择，可以提高检索的准确性和相关性。
4. 数据增强和样本平衡：在训练LLM模型时，可以增加垂直领域的样本数据，以增加模型对垂直领域的理解和表达能力。同时，要注意样本的平衡，确保训练数据中包含各个垂直领域的典型对话场景，避免偏向某个特定领域。
5. 引入外部知识库：在垂直领域的对话中，可以结合外部的领域知识库，如专业词典、行业标准等，来提供更准确的答案和解决方案。通过与外部知识库的结合，可以弥补LLM模型和向量库在垂直领域中的不足。
6. 收集用户反馈和迭代优化：通过收集用户的反馈信息，了解用户对对话系统的需求和期望，并根据反馈进行迭代优化。持续改进和优化是提高垂直领域对话效果的关键。

总之，通过领域特定训练、增加领域知识、优化检索算法、数据增强和样本平衡、引入外部知识库以及收集用户反馈和迭代优化等方法，可以改进基于LLM和向量库的文档对话在垂直领域中的表现。这些方法可以根据具体情况灵活应用，以提高对话系统的准确性和适应性。

3. 💡 痛点3：langchain 内置 问答分句效果不佳问题

如果您在使用Langchain内置的问答分句功能时发现效果不佳，可以尝试以下方法来改善：

1. 调整输入：检查输入的文本是否符合预期的格式和结构。确保输入的句子和段落之间有明确的分隔符，如句号、问号或换行符。如果输入的文本结构不清晰，可能会导致分句效果不佳。
2. 引入标点符号：在文本中适当地引入标点符号，如句号、问号或感叹号，以帮助模型更好地理解句子的边界。标点符号可以提供明确的分句信号，有助于改善分句的准确性。
3. 使用自定义规则：针对特定的文本类型或语言，可以使用自定义规则来分句。例如，可以编写正则表达式或使用特定的分句库来处理特定的分句需求。这样可以更好地适应特定的语言和文本结构。
4. 结合其他工具：除了Langchain内置的问答分句功能，还可以结合其他分句工具或库来处理文本。例如，NLTK、spaCy等自然语言处理工具包中提供了强大的分句功能，可以与Langchain一起使用，以获得更好的分句效果。
5. 使用上下文信息：如果上下文信息可用，可以利用上下文信息来辅助分句。例如，可以根据上下文中的语境和语义信息来判断句子的边界，从而提高分句的准确性。
6. 收集反馈和调整模型：如果您发现Langchain内置的问答分句功能在特定场景下效果不佳，可以收集用户反馈，并根据反馈进行模型调整和改进。通过不断优化模型，可以逐渐改善分句效果。

总之，通过调整输入、引入标点符号、使用自定义规则、结合其他工具、使用上下文信息以及收集反馈和调整模型等方法，可以改善Langchain内置的问答分句效果。这些方法可以根据具体情况灵活使用，以提高分句的准确性和效果。

4. 💡 痛点4：如何 尽可能召回与query相关的Document 问题

要尽可能召回与query相关的Document，可以采取以下方法：

1. 建立索引：将Document集合建立索引，以便能够快速检索和匹配相关的Document。可以使用搜索引擎或专业的信息检索工具，如Elasticsearch、Solr等。
2. 关键词匹配：通过对query和Document中的关键词进行匹配，筛选出包含相关关键词的Document。可以使用TF-IDF、BM25等算法来计算关键词的重要性和匹配程度。
3. 向量化表示：将query和Document转化为向量表示，通过计算它们之间的相似度来判断相关性。可以使用词嵌入模型（如Word2Vec、GloVe）或深度学习模型（如BERT、ELMo）来获取向量表示。
4. 上下文建模：考虑上下文信息，如query的前后文、Document的上下文等，以更准确地判断相关性。可以使用上下文编码器或注意力机制来捕捉上下文信息。
5. 扩展查询：根据query的特点，进行查询扩展，引入相关的同义词、近义词、词根变化等，以扩大相关Document的召回范围。
6. 语义匹配：使用语义匹配模型，如Siamese网络、BERT等，来计算query和Document之间的语义相似度，以更准确地判断相关性。
7. 实时反馈：利用用户的反馈信息，如点击、收藏、评分等，来优化召回结果。通过监控用户行为，不断调整和优化召回算法，提升相关Document的召回率。
8. 多模态信息利用：如果有可用的多模态信息，如图像、视频等，可以将其整合到召回模型中，以提供更丰富、准确的相关Document。通过多模态信息的利用，可以增强召回模型的表达能力和准确性。

总之，通过建立索引、关键词匹配、向量化表示、上下文建模、查询扩展、语义匹配、实时反馈和多模态信息利用等方法，可以尽可能召回与query相关的Document。这些方法可以单独使用，也可以结合起来，以提高召回的准确性和覆盖率。

5. 💡 痛点5：如何让LLM基于query和context得到高质量的response

要让LLM基于query和context得到高质量的response，可以采取以下方法：

1. 数据准备：准备大量高质量的训练数据，包括query、context和对应的高质量response。确保数据的多样性和覆盖性，以提供更好的训练样本。
2. 模型架构：选择合适的模型架构，如Transformer等，以便提取query和context中的重要信息，并生成相应的高质量response。确保模型具有足够的容量和复杂性，以适应各种复杂的查询和上下文。
3. 微调和优化：使用预训练的模型作为起点，通过在特定任务上进行微调和优化，使模型能够更好地理解query和context，并生成更准确、连贯的response。可以使用基于强化学习的方法，如强化对抗学习，来进一步提高模型的表现。
4. 上下文建模：在LLM中，上下文对于生成高质量的response非常重要。确保模型能够准确地理解和利用上下文信息，以生成与之相关的response。可以使用一些技术，如注意力机制和上下文编码器，来帮助模型更好地建模上下文。
5. 评估和反馈：定期评估模型的性能，使用一些评估指标，如BLEU、ROUGE等，来衡量生成的response的质量。根据评估结果，及时调整和改进模型的训练策略和参数设置。同时，收集用户反馈和意见，以便进一步改进模型的性能。
6. 多模态信息利用：如果有可用的多模态信息，如图像、视频等，可以将其整合到LLM中，以提供更丰富、准确的response。利用多模态信息可以增强模型的理解能力和表达能力，从而生成更高质量的response。
7. 引入外部知识和资源：为了提高LLM的质量，可以引入外部知识和资源，如知识图谱、预训练的语言模型等。利用这些资源可以帮助模型更好地理解和回答query，从而生成更高质量的response。

总之，通过合适的数据准备、模型架构选择、微调和优化、上下文建模、评估和反馈、多模态信息利用以及引入外部知识和资源等方法，可以帮助LLM基于query和context得到高质量的response。

3. 基于LLM+向量库的文档对话 工程示例面

1. 本地知识库问答系统（Langchain-chatGLM）

1. 避坑记录