석사학위논문

RAG 기법을 활용한 LLM 서비스 품질 향상을 위한 데이터 Chunking 전략에 대한 연구

고려대학교컴퓨터정보통신대학원

인공지능융합학과

김종철

2024년 2월

임희석 교수지도

석사학위논문

RAG 기법을 활용한 LLM 서비스 품질 향상을 위한 데이터 Chunking 전략에 대한 연구

이 논문을 공학 석사학위 논문으로 제출함

2023년 10월

고려대학교컴퓨터정보통신대학원

인공지능융합학과

김종철(인)

김종철의 석사학위논문 심사를 완료함

2023 년 12 월

위원장 임 희 석 (인)

위 원 김 현 철 (인)

위 원 강 재 우 (인)

RAG 기법을 활용한 LLM 서비스 품질 향상을 위한 데이터 Chunking 전략에 대한 연구

김 종 철

인 공 지 능 융 합 학 과

지도교수: 임 희 석

초록 (국문)

자연어 처리(NLP) 분야에서 GPT 와 같은 언어 모델들이 큰 성과를 거두고 있으나, 대규모 데이터에 의존하는 훈련 방식은 특정 도메인에 대한 정보 제공에 한계를 가진다. RAG(Retrieval-Augmented Generation) 기법은 이 한계를 극복하기 위해 문서 검색과 응답 생성을 결합하여 더 정확한 정보 제공을 목표로 한다. 본 연구는 문서 데이터의 청킹 전략, 즉 데이터를 처리 단위로 나누는 방법이 RAG 기법의 성능에 미치는 영향을 집중적으로 분석한다. 실험을 통해 다양한 청크 크기의 조합이 검색 및 생성 성능에 어떤 영향을 미치는지 평가하였다. 단일 청크 사이즈보다 조합된 복합 청크 사이즈를 사용하는 것이 일반적으로 검색 및 답변 성능 향상에 기여할 수 있음을 확인하였다. 다양한 크기의 청크를 혼합함으로써 모델이 더 다양한 맥락에서 정보를 학습하고, 이를 통해 더 정확한 임베딩을 생성할 수 있기 때문으로 해석될 수 있다. 복합 청크 분할 방식을 사용하여 LLM 서비스의 품질 향상을 위한 최적의 청킹 전략을 제안한다.

중심어: LLM, RAG, Chunking, GPT, NLP

Research on Data Chunking Strategies for Enhancing LLM Service Quality Using the RAG Technique

by Jongcheol Kim

Department of Applied Artificial Intelligence

under the supervision of Professor Heuiseok Lim

ABSTRACT

In the field of Natural Language Processing (NLP), language models like GPT have achieved significant success, yet the training approach that relies on large-scale data has limitations in providing information for specific domains. The Retrieval-Augmented Generation (RAG) technique aims to overcome these limitations by combining document retrieval and response generation to aim for more accurate information provision. This study focuses on analyzing the impact of document data chunking strategies, that is, the method of dividing data into processing units, on the performance of the RAG technique. Through experimentation, we evaluated how combinations of various chunk sizes affect search and generation performance. It was found that using combined composite chunk sizes generally contributes to improved search and answer performance compared to single chunk sizes. This can be interpreted as allowing the model to learn information in a more diverse context and thereby generate more accurate embeddings. We propose an optimal chunking strategy for improving the quality of LLM services by utilizing a composite chunk division approach.

Keywords: LLM, RAG, Chunking, GPT, NLP

목차

표 목차

그림 목차

1 장. 서론

1.1 연구 배경

최근 몇 년 동안 언어 모델은 놀라운 발전을 이루었다. GPT, BERT 와 같은 모델들은 다양한 자연어 처리 작업에서 뛰어난 성능을 보여주었다. 이러한 모델들은 클라우드 기반 서비스로 제공되며, 다양한 애플리케이션에서 활용되고 있다.

특히 GPT 와 같은 모델들의 발전은 다양한 분야에서의 텍스트 생성 및 정보 제공에 큰 기여를 하고 있다. 이러한 모델들은 사용자와 기계 간의 원활한 소통을 가능하게 하며, 다양한 산업 분야에 혁신을 가져오고 있다. 그러나 모델들의 대규모 훈련 데이터의 한계로 인해 특정 도메인이나 조직의 특화된 정보 제공에 한계가 있으며, 때로는 부정확한 정보를 생성하는 문제점이 있다.

이러한 문제를 해결하기 위한 전략 중 하나로 RAG (Retrieval-Augmented Generation) 기법이 주목받고 있다. RAG 는 문서 검색과 생성을 결합하여 사용자의 질문에 대한 답변을 제공하는 방식으로, 대규모 문서 데이터베이스에서 관련 정보를 검색하고, 그 정보를 기반으로 응답을 생성한다. 이 방식은 기존 언어 모델만을 활용하는 것보다 더욱 다양하고 정확한 정보 제공이 가능하다. 그러나 RAG 의 성능은 문서 데이터의 구조와 임베딩(Embedding) 방식에 크게 의존한다.

문서 데이터 임베딩하기 위해서 문장(Sentence), 단락(Paragraph), 장(Page) 단위로 분리하여 성능을 평가해 보았는데 이 방식은 여러 가지 문제점이 있었다. 문장이나 단락 단위로 분리하기 위해서는 문서 데이터에 마침표(.), 줄 바꿈 등 명확한 구분자가 필요하지만 대부분의 문서에는 해당 구분자가 없거나 규칙이 다른 경우가 많다.

또한 문서의 장 단위로 분리하는 경우, 많은 내용을 포함하고 있기 때문에 답변의 품질은 좋아질 수 있지만 불필요한 정보까지 포함되어 부정확한 답변을 생성하는 문제가 발생한다. 이는 허위정보를 생성하는 것을 의미한다. 이러한 이유로 데이터 청킹(Chunking) 전략은 RAG 기법의 성능에 결정적인 영향을 미친다.

청크의 핵심은 문서가 청크(Chunk)로 나누어진 이후에도 각 청크가 문서에서 해당하는 부분의 내용과 맥락을 그대로 보존하는 것이다. 잘못된 청킹은 시스템에 전달된 질문으로부터 원하는 내용을 불러오는 것이 힘들다. 그 이유는 텍스트의 일부가 잘리거나 관련 없는 내용이 추가되면 그 의미가 크게 달라지며, 이는 유사도 기반의 검색(retrieval)에 크게 영향을 미치기 때문이다. 그리고 청크를 제대로 가지고 왔다 하더라도 청크가 완전하지 않을 경우, LLM(Large Language Model)은 이를 조건으로 불완전한 대답을 만들어낼 것이기 때문에 생성(generate) 성능에도 영향을 미친다. 따라서 청킹 전략(chunking strategy)은 RAG 시스템을 구축하는 데 있어 매우 중요한 요소이다.

본 연구에서는 데이터의 청킹 전략에 중점을 둔다. 일반적으로 하나의 청크 사이즈(Chunk Size)만을 고려하는 경우가 많으나, 본 연구는 단일 청크 사이즈가 아닌 복합 청크 사이즈를 도입하는 방식을 제안한다. 이를 통해 세부 정보부터 넓은 맥락의 정보까지 다양한 범위의 정보를 포착하고 활용할 수 있을 것으로 기대된다. 이러한 접근은 서비스의 품질과 정확성을 향상시키는 데 중요한 역할을 할 것으로 예상된다.

1.2 연구 목적

본 연구는 RAG (Retrieval-Augmented Generation) 기법을 활용한 LLM 서비스의 품질 향상을 위한 데이터 청킹 전략의 효과를 깊게 탐구하는 것을 주요 목적으로 한다.

특히, 다양한 청크 크기가 RAG 기법의 성능에 미치는 영향을 중심으로 연구를

진행하며, 이를 통해 최적의 청크 크기와 데이터 청킹 전략을 도출하고자 한다.

이 연구를 통해, 여러 사이즈의 청크를 벡터 데이터베이스에 저장하고, 그 결과를 바탕으로 서비스의 품질을 평가하여 LLM 서비스의 품질 향상에 기여할 수 있는 최적의 청크 크기와 전략을 제안하고자 한다.

2 장. 배경 지식

2.1 LLM (Large Language Model) 정의

거대 언어 모델 (LLM)은 대규모 언어 데이터에 기반하여 학습되는 모델로, 딥 러닝 알고리즘과 통계적 방법론의 통합을 통해 자연어 처리 (NLP)의 다양한 작업에 활용되고 있다. 이러한 모델은 광범위한 언어 데이터를 통해 문장의 구조, 문법, 그리고 의미를 깊이 있게 학습하며, 결과적으로 텍스트의 이해 및 생성 능력을 향상시킨다.

LLM 의 주요 응용 분야 중 하나는 주어진 문맥에 기반하여 후속 단어나 문장을 예측하는 작업이다. 이는 문장 내의 단어들 간의 관계와 문맥을 상세히 분석하여 이루어지며, 이러한 기능은 기계 번역, 텍스트 요약, 자동 작문, 질의 응답 등의 NLP 작업에 광범위하게 적용되고 있다.

현대의 LLM 은 GPT, BERT 와 같은 다양한 아키텍처를 포함하고 있으며, 이들 모델은 수천억 개 이상의 매개변수를 포함할 수 있다. 최근의 연구 동향은 대규모 훈련 데이터와 고도화된 모델 아키텍처를 활용하여 언어의 미세한 이해와 정교한 생성 능력을 추구하는

방향으로 진행되고 있다.

2017 년 구글이 Transformer 모델을 발표함으로써 순차 데이터 학습에 대한 새로운 패러다임을 제시하였다. 이어 2018 년에는 구글의 BERT 모델이 NLP 벤치마크에서 뛰어난 성능을 보여주며 주목받게 되었다. 2020 년에는 OpenAI 와 존스홉킨스 대학의 연구팀이 GPT-3 를 발표하였고, 1,750 억 개의 매개변수(Parameter)를 포함하는 이 모델은 LLM 분야의 주요성을 대중에게 알리게 되었다. 문장이나 문서의 일부를 입력으로 받아 빈칸이나 다음 단어를 예측하는 언어 모델링 태스크에서 뛰어난 성능을 보여주었다 . 2021 년과 2022 년에는 각각 Github Copilot 과 ChatGPT 의 발표를 통해 LLM 의 접근성과 활용도가 더욱 확대되었다.

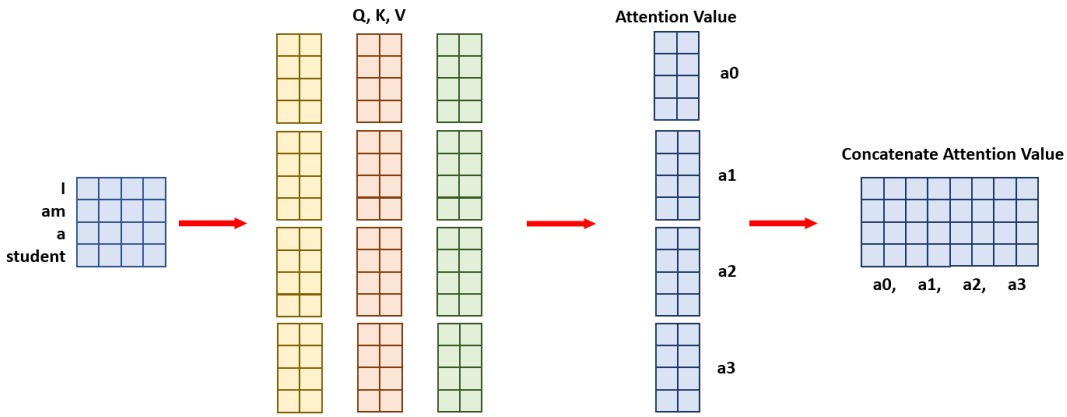
2.2 LLM 동작원리

거대 언어 모델 (LLM)은 주로 트랜스포머(Transformer) 아키텍처에 기반하여 작동한다. 이러한 모델은 대규모 코퍼스를 활용하여 토큰의 임베딩 벡터를 학습하며, 이 임베딩 벡터들을 집합하여 문서 전체를 표현하는 벡터를 생성한다. 이렇게 학습된 벡터 표현은 문서 간의 유사도 측정, 분류 등의 다양한 NLP 작업에 활용될 수 있다.

LLM 은 딥러닝 방식으로 방대한 양을 사전학습(pre-trained)한 전이학습(transfer) 모델이다. 따라서 인간의 두뇌가 학습하는 방식과 유사하다. LLM 은 문장에서 가장 자연스러운 단어 시퀀스를 찾아내는 딥러닝 모델이며, 문장 속에서 이전 단어들이 주어졌을 때 다음 단어를 예측하거나, 주어진 양쪽의 단어들 사이에서 가운데 단어를 예측하도록 하는 등의 방식으로 작동한다.

GPT-3.5 와 같은 모델은 학습된 파라미터를 활용하여 주어진 입력에 대한 응답을 생성한다. 구체적으로, 입력이 주어지면 해당 입력은 학습 과정에서 결정된 토큰 단위로

분리되며, 이 토큰들은 기존에 학습된 문장 구조와 문법 규칙을 바탕으로 처리된다. 트랜스포머 아키텍처 내의 멀티 헤드 어텐션 (Multi-head attention) 메커니즘을 통해 각 토큰의 중요도가 계산되며, 이를 바탕으로 다음 토큰의 확률 분포가 예측된다. 이 과정을 반복하여 전체 응답이 생성된다.



[그림 1] Multi-head Attention 메커니즘

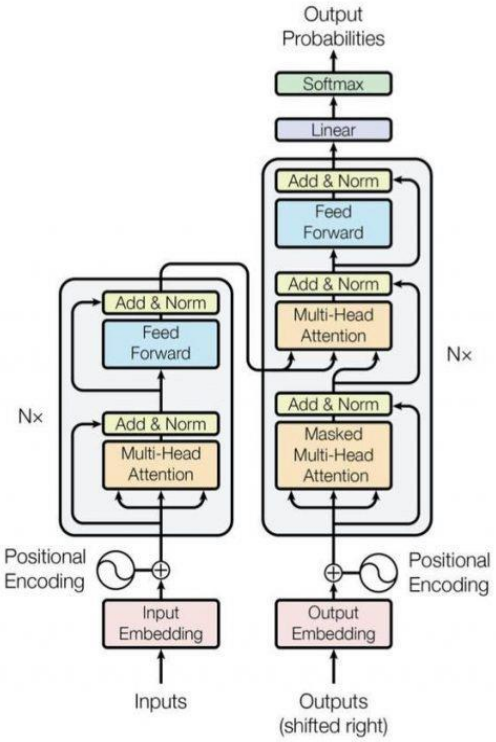
head 의 수가 4 개이므로 각 연산과정이 4 분의 1 만큼만 필요하다는 의미이다. 그렇기때문에 크기가 [4x8]의 Query, Key, Value 를 4 등분하여 [4x2]로 만든다. 그러면 Attention Value 는 [4x2]가 된다. 이 Attention Value 들을 마지막에 concatenate 를 시켜 위 그림과 같이 크기가 [4x8]이 되어 일반적인 Attention 메커니즘의 결과값과 동일하게 된다.

토큰은 문장이나 단어를 분석하기 위해 세분화된 최소 단위로, 특정 언어에서는 토큰이 여러 알파벳으로 구성될 수 있다. Transformer 아키텍처는 RNN 이나 CNN 과는 다르게 self-attention 메커니즘을 중심으로 동작하며, 이를 통해 문장 내의 토큰 간의 의존성을 효과적으로 모델링한다. Transformer 의 핵심 구성 요소는 Encoder 와 Decoder 로, Encoder 는 입력 시퀀스를 처리하고, Decoder 는 이를 바탕으로 출력 시퀀스를 생성한다.

Transformer 의 self-attention 메커니즘은 입력 시퀀스 내의 토큰 간의 관계를

모델링하는 데 사용된다. 이 메커니즘은 Query, Key, Value 벡터를 생성하고, 이들 벡터를 활용하여 각 토큰의 중요도를 계산한다. Multi-Head Attention 은 이러한 self-attention 메커니즘을 여러 개 병렬로 수행하여 다양한 관점에서의 토큰 간의 관계를 파악한다.

LLM 의 크기는 그 모델이 포함하는 매개변수의 수로 표현된다. 예를 들어, GPT-3 175B 모델은 약 1,750 억 개의 매개변수를 포함하며, 매개변수는 모델의 학습과정에서 최적화된다.



[그림 2] Transformer 구조

이러한 인공 신경망 기반의 언어 모델들은 방대한 양의 데이터를 학습하여 인간과 같은 자연스러운 문장을 생성한다. 정교해진 GPT 는 보다 더욱 세밀하게 사람의 의도를 이해하고, 적절한 답을 할 줄 알며, 심지어는 사람처럼 말하는 법까지 배우고 있다.

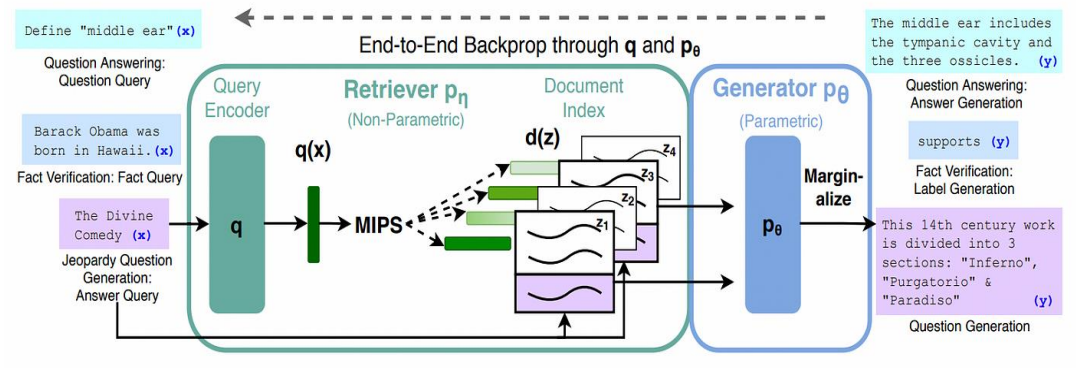
2.3 RAG (Retrieval-Augmented Generation) 정의

Retrieval Augmented Generation (RAG)는 최근 주목받는 기술로, 미리 학습된 대규모 언어 모델 (LLM)과 외부 데이터를 결합하여 응답 생성에 활용하는 방식을 제시한다. RAG 의 핵심 원리는 기존의 LLM 이 독립적으로 응답을 생성하는 대신, 외부 데이터를 검색(retrieval)하여 해당 데이터를 문맥으로 활용하는 것이다.

이 접근법은 임베딩 모델을 통해 텍스트 데이터를 벡터 공간에 표현하는 임베딩을 생성하는 과정으로 시작된다. 이렇게 생성된 임베딩은 벡터 데이터베이스에 저장되며, 사용자의 질문이나 입력이 주어질 때, 시맨틱 검색을 통해 관련된 정보를 검색한다. 이 검색 방식은 전통적인 텍스트 기반 검색과는 다르게, 임베딩을 활용하여 의미론적으로 유사한 정보를 검색하는 것이 특징이다.

2.4 RAG 의 작동 원리

RAG 는 언어 모델에게 파라미터에 내포된 지식 이외의 정보를 전달함으로써, 질문에 대한 사실 기반의 대답을 생성하게 만드는 기술이다. 최근 LLM 을 이용한 다양한 기법의 RAG 가 등장하고 있지만, 기본적으로 RAG 시스템은 크게 두 부분, 정보를 불러오는 retrieval 파트와 불러온 정보로부터 답을 생성하는 generate 파트로 나뉘어진다.



[그림 3] RAG 프레임워크

해당하는 query text 가 입력되면 인코딩 결과를 DPR 프레임워크로 구성된 Retriever 에서 관련 top-k 문단들을 먼저 찾아내고, 문단과 질문 정보를 활용하여 언어 모델 Generator 에서 해당하는 답변 문장을 생성해내는 것이 RAG 프레임워크의 구성이다.

RAG 는 미세 조정이나 추가 학습과 같은 고비용의 방법 대신, 실시간으로 외부 데이터를 활용하여 LLM 의 성능을 향상시키는 저비용의 방법을 제공한다. 이로 인해, 기업은 최신 데이터를 활용하여 LLM 을 실시간으로 보완하거나 안내할 수 있으며, 이는 특히 도메인이 지속적으로 변화하는 환경에서 매우 유용하다.

RAG 는 LLM 의 추론 능력과 외부 데이터의 정보를 결합하여 더욱 풍부하고 정확한 응답을 생성하는 기술로, 다양한 도메인에서의 활용 가능성이 높다.

2.5 RAG 의 장점 및 한계

RAG 는 컴퓨터 시스템에서 정보를 합성하고 응답을 생성하는 데 있어 여러 중요한 이점을 갖고 있다. 이 모델은 특히 다양한 정보와 지식을 기반으로 한 응답을 생성할 수 있는 능력이 뛰어나며, 이는 방대한 문서 데이터베이스를 활용하기 때문이다. 또한, 검색과

생성을 통합하는 과정을 통해 사용자의 질문에 정확한 답변을 제공할 수 있으며, 다양한 도메인과 언어에 적용할 수 있어 유연성도 매우 높다.

그러나 RAG 모델도 몇 가지 제약 사항을 갖고 있다. 먼저, 대규모 데이터베이스를 검색하는 과정이 계산 복잡성을 증가시킬 수 있다. 이는 고성능의 컴퓨팅 리소스를 필요로 한다. 또한, 모델의 효과는 고품질의 데이터베이스에 크게 의존하고 있기때문에 데이터의 질이 모델 성능에 직접적인 영향을 미친다. 마지막으로, 데이터의 청크 크기에 따라 모델의 성능에 큰 차이가 발생할 수 있는데, 이는 청크 크기가 응답 생성의 정확도와 효율성에 영향을 미치기 때문이다. 따라서 최적의 성능을 위해 적절한 청크 크기를 선정하는 것이 중요하다.

2.6 데이터 청킹의 중요성

데이터 청킹은 대규모의 데이터를 효과적으로 관리하고 처리하기 위한 핵심 전략으로 데이터를 작은 단위로 분할하는 과정을 의미한다. 이러한 청크는 데이터의 처리, 저장, 검색 등의 작업을 효율적으로 수행하기 위해 사용된다.

특히, RAG 와 같은 기법에서는 데이터의 청크 크기가 성능에 큰 영향을 미칠 수 있다.

RAG 는 최근 다양한 기업 및 연구자들에게 주목받는 기술로, 대규모 언어 모델과 외부 데이터의 결합을 통해 향상된 응답 생성 능력을 추구한다. 그러나 실제 프로덕트 수준의 응용에서는 RAG 의 성능이 기대에 못 미치는 경우가 종종 관찰된다. 이러한 문제의 원인 중 하나로, 문서의 청킹 전략 (Chunking Strategy)의 중요성이 강조된다.

성능이 뛰어난 RAG 는 다음의 조건을 충족시켜야 한다. 첫째, 문서를 청크로 분할할 때, 각 청크가 원본 문서의 정보와 맥락을 손상 없이 보존해야 한다. 둘째, 사용자의 질문에

대한 적절한 근거 정보를 효과적으로 검색해야 한다. 셋째, 검색된 정보들을 적절하게 결합하여 응답을 생성해야 한다. 특히, 청크를 분할하는 전략은 RAG 의 성능에 결정적인 영향을 미친다. 청킹의 목적은 문서의 일부분을 독립적인 단위로 나누는 것이지만, 이 과정에서 원본 문서의 의미나 맥락이 손상되면 해당 청크는 검색과 생성 과정에서 문제를 야기할 수 있다.

전문적인 도메인에서는 청크의 완전성과 정확성이 매우 중요하다. 잘못된 청킹은 두 가지 주요 문제를 초래한다. 첫째, 의미나 맥락이 손상된 청크는 유사도 기반의 검색에서 정확한 결과를 가져오기 어렵다. 둘째, 검색된 청크가 완전하지 않거나 잘못된 정보를 포함하고 있다면, LLM 은 이를 기반으로 불완전하거나 잘못된 응답을 생성할 수 있다. 청킹 전략은 RAG 시스템의 성능과 효율성을 결정짓는 핵심 요소로, 이를 통해 LLM 의 응답 생성 능력을 극대화할 수 있다.

3 장. 실험

3.1 데이터 셋

본 연구에서는 RAG 기법의 성능 평가를 위해 다양한 도메인의 텍스트 데이터를 수집하였다. PDF 문서 50 건를 활용하였고 문서의 양식이 표준화되지 않은 문서를 선정하였다. 비표준 문서를 실험대상으로 선정한 이유로는 사내에서 생성되는 대부분의 문서는 표준화된 양식으로 작성된 것 보다 자유 형식으로 작성된 문서의 비율이 높기 때문이다.

표 1. 수집 데이터 셋

데이터 종류	문서 수	문장 수	데이터 형식
공정거래	10	5,382	pdf
법령	10	2,600	pdf
기술 문서	30	3,188	pdf

전반적으로 데이터 전처리는 문서 특성에 따라 다르게 진행되기 때문에 최소한의 전처리만 수행하였다. 데이터 전처리로는 불필요한 특수문자를 제거하였으며 민감한 단어는 변환하여 보안 문제를 차단하였다. 그리고 단일 크기와 복합 크기를 설정하여 아래와 같은 사이즈로 청크를 분할하였다.

표 2. 청크 사이즈

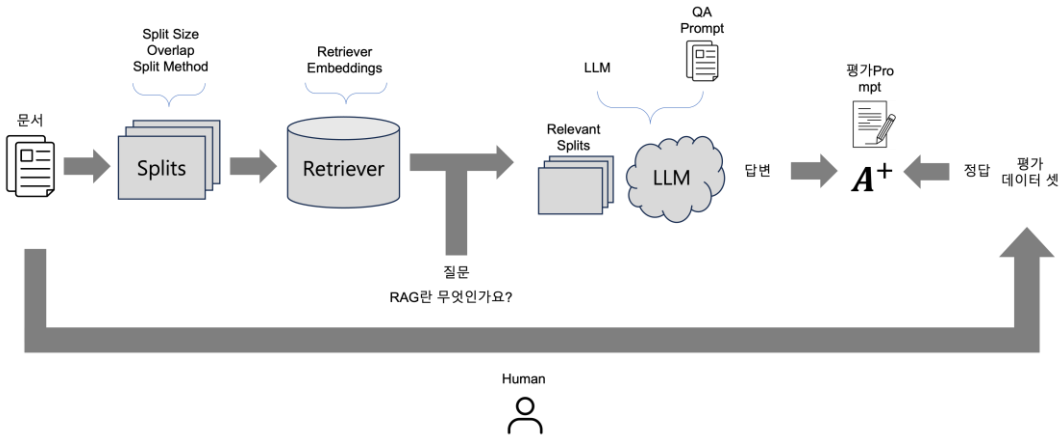
종류	사이즈
단일	128, 256, 512, 1024
복합	[128, 512], [256, 512], [128, 1024], [256, 1024], [512, 1024]

또한, LLM 서비스의 성능을 평가하기 위해 수집한 데이터에서 평가 데이터셋을 생성하였다.

3.2 실험 설계

본 실험은 다양한 청크 사이즈를 설정하여 청크 사이즈별로 검색 성능 및 응답 정확도를 평가하였다. 실험을 위해서 LangChain LLM Framework 를 사용하여 구조화되지 않은 원시 데이터를 QA(Question-Answering) 체인으로 변환하는 Data

Loading -> Text Splitting -> Storage -> Retrieval -> Generation -> Evaluation 의 파이프라인을 구성하였다.



[그림 4] QA 파이프라인

Data Loading 단계에서는 다양한 형식과 소스의 데이터에 액세스하고 표준화된 형식으로 변환하는 세부 사항을 처리한다. 원본 데이터가 관리 가능한 조각으로 나누어지고 데이터는 문서, 데이터베이스 등 다양한 출처에서 가져올 수 있다.

Text Splitting 단계에서 원본 데이터는 일반 텍스트로 변환되며, 이 텍스트는 더 작은 조각, 즉 청크로 나누어진다.

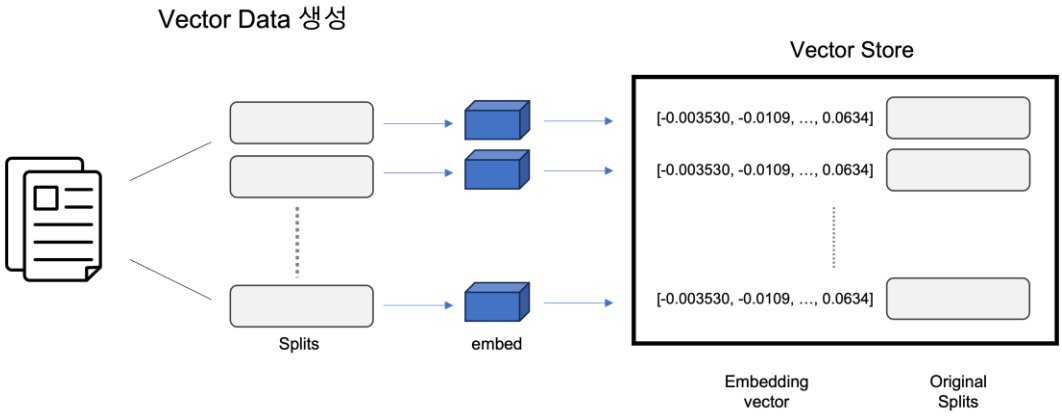
텍스트를 청크 단위로 분할하기 위해 LangChain 의 RecursiveCharacterTextSplitter 를 사용하였고, 이후 각 청크는 임베딩 모델을 통해 벡터 공간에 표현되는 임베딩으로 변환한다.

데이터를 벡터 공간에 임베딩하기 위해 OpenAI 에서 제공하는 text-embedding-ada-002 모델을 활용하였다.

표 3. 데이터 청크 분할 비교

사이즈	데이터	토큰 수	글자 수
128	공정거래준수 편람 본편람은 포트리스의 지적자산으로서 포트리스의 승인없는 무단복제 및배포를 금지합니다 Compliance Program2023 마케팅부문 공정거래준수 편람 Compliance Program 2023 01 개정판	99	128
256	공정거래준수 편람 본편람은 포트리스의 지적자산으로서 포트리스의 승인없는 무단복제 및배포를 금지합니다 Compliance Program2023 마케팅부문 공정거래준수 편람 Compliance Program 2023 01 개정판 본편람은 포트리스의 지적자산으로서 포트리스의 승인없는 무단복제 및배포를 금지합니다 마케팅부문 공정거래준수 편람개정판을 내면서 현재 수많은 글로벌 기업들은 단기적인 수익창출의 관점에서 벗어나 투명하고 공정한 기업활동에	233	253

Storage 단계는 데이터를 쉽게 액세스할 수 있는 형식으로 저장해야 하므로 벡터 저장 및 임베딩은 텍스트 분할 이후에 발생한다. 임베딩은 텍스트 조각을 가져와 텍스트의 숫자 표현을 만든다. 따라서 의미상 유사한 내용을 가진 텍스트는 임베딩 공간에서 유사한 벡터를 갖게 된다. 따라서 임베딩(벡터)을 비교하고 유사한 텍스트를 찾을 수 있다. 데이터를 청크 단위로 분할하고 임베딩으로 변환 후 벡터 데이터베이스에 저장한다. 임베딩 정보와 원본 데이터와의 연결 정보를 포함하는 메타데이터와 함께 저장된다. 이 메타데이터는 LLM 이 응답을 생성할 때 참조나 인용을 생성하는 데 활용된다. 이번 실험에서 사용한 벡터 데이터베이스는 FAISS 를 활용하였다.



[그림 5] 벡터 데이터베이스 임베딩 저장

Retrieval 단계는 평가를 위해 생성한 평가데이터셋을 사용하여 입력 질문과 유사한 임베딩이 포함된 데이터를 검색한다. 정확한 평가를 위해 실무자가 직접 문제와 정답셋을 아래와 같은 형태로 100 문항을 생성하였고, 평가를 자동화 하기 위해서 json 형태로 가공하여 활용하였다.

표 4. 평가 데이터셋

문제	정답
가격 변동 또는 설계 변경에 따른 하도급 지급금의 조정 방법과 시기는 어떻게 되나요?	하도급 지급금은 발주자로부터 조정받은 날로부터 30 일 이내에 조정되어야 하며, 고객으로부터 받은 지급금은 수령일로부터 15 일 이내에 하도급업자에게 지급되어야 합니다.
변압기에 영향을 미치는 부하의 특성은 무엇인가요?	변압기는 송배전 계통에 직렬로 연결되어 전력을 전달하는 기기이므로 전원측과 부하측 양쪽의 설비 기능을 저해하지 않고 본래의 역할을 충분히 수행할 수 있는 것이여야 합니다.

Generation 단계는 질문과 벡터 데이터베이스에서 검색된 결과를 기반으로 LLM 을 통해서 답변을 생성한다. 질문과 유사한 내용을 벡터 데이터베이스에서 Similarity Search 를 통해 top-k 를 선정하고, 선정된 데이터를 포함하여 gpt-3.5-turbo-16k, gpt-4

모델을 활용하여 답변을 생성할 수 있도록 구성 하였다.

Evaluation 단계에서는 생성된 답변을 기준으로 응답 시간, 검색에 대한 결과 정확성, 답변의 정확성을 평가하였다. 각 성능에 대한 평가는 gpt-3.5-turbo-16k 모델을 사용했고, 평가를 위해 아래와 같이 프롬프트를 생성하였다.

표 5. 평가 프롬프트

평가 유형	프롬프트
답변에 대한 평가	""" 당신은 퀴즈를 채점하는 선생님입니다. 문제, 학생의 답변, 그리고 정답이 주어지며, 학생의 답변을 '정답' 또는 '오답'으로 평가해야 합니다. 예시 형식: 문제: 여기에 문제 작성 학생의 답변: 여기에 학생의 답변 작성 정답: 여기에 정답 작성 평가: '정답' 또는 '오답'으로 작성 학생의 답변을 오로지 사실적 정확성에 기반하여 평가하세요. 학생의 답변이 정답보다 더 많은 정보를 포함하고 있어도 괜찮습니다. 단, 그 정보가 서로 모순되는 내용을 포함하고 있지 않아야 합니다. 문제: {query} 학생의 답변: {result} 정답: {answer} 평가: """
검색에 대한 평가	""" 주어진 문제: {query} 다음으로 검색된 내용이 질문과 관련이 있는지 결정하세요: {result} 다음 형식으로 답하세요:

" 참 또는 거짓."
그리고 왜 그것이 정답인 {answer}을 지지하거나 지지하지 않는지 설명하세요.
"""

평가지표는 정확도(Accuracy)를 활용하여 지연율(Latency), 검색 정확도, 답변 정확도를 평가 하였다. 정확도는 올바르게 예측된 데이터의 수를 전체 데이터의 수로 나눈 값이며 수식은 다음과 같다.

$$accuracy(\mathcal{Y}, \hat{\mathcal{Y}}) = \frac{1}{n_{samples}} \sum_{i=0}^{n_{smaples}-1} 1(\hat{\mathcal{Y}}_i = \mathcal{Y}_j) \quad (1)$$

그리고 BERTScore 를 활용하여 청크 사이즈 별 P_{BERT}, R_{BERT}, F_{BERT} 를 계산하여 평가하였다. BERTScore 점수는 정답 요약문과 생성 요약문을 BERT 에 입력하여 문맥 벡터를 구한 후 이를 기반으로 코사인 유사도를 계산해 점수를 구하는 방식이다. BERTScore 는 x 의 각 토큰을 xˆ 의 토큰과 매치 시켜 리콜(Recall)을 계산하고, xˆ 의 각 토큰을 x 의 토큰과 매치 시켜 정밀도(precision)를 계산한다. 또한 매칭 유사도 점수를 극대화하기 위해 탐욕적 매칭(greedy matching)을 사용한다. 여기서 각 토큰은 다른 문장의 가장 유사한 토큰과 매치된다. 정밀도와 리콜을 결합하여 F1 척도(F1 measure)를 계산한다. 참조 x 와 후보 xˆ 에 대해, 리콜, 정밀도, F1 점수(F1 Score)를 계산하는 수식은 다음과 같다.

$$R_{BERT} = \frac{1}{|x|} \sum_{x_i \in x} \max_{\hat{x}_j \in \hat{x}} \mathrm{X}_i^{\mathrm{T}} \hat{\mathrm{X}}_j \quad (2)$$

$$P_{BERT} = \frac{1}{|\hat{x}|} \sum_{\hat{x}_j \in \hat{x}} \max_{x_i \in x} \mathrm{X}_i^{\mathrm{T}} \hat{\mathrm{X}}_j \quad (3)$$

$$F_{BERT} = 2\frac{P_{BERT} \cdot R_{BERT}}{P_{BERT} + R_{BERT}} \quad (4)$$

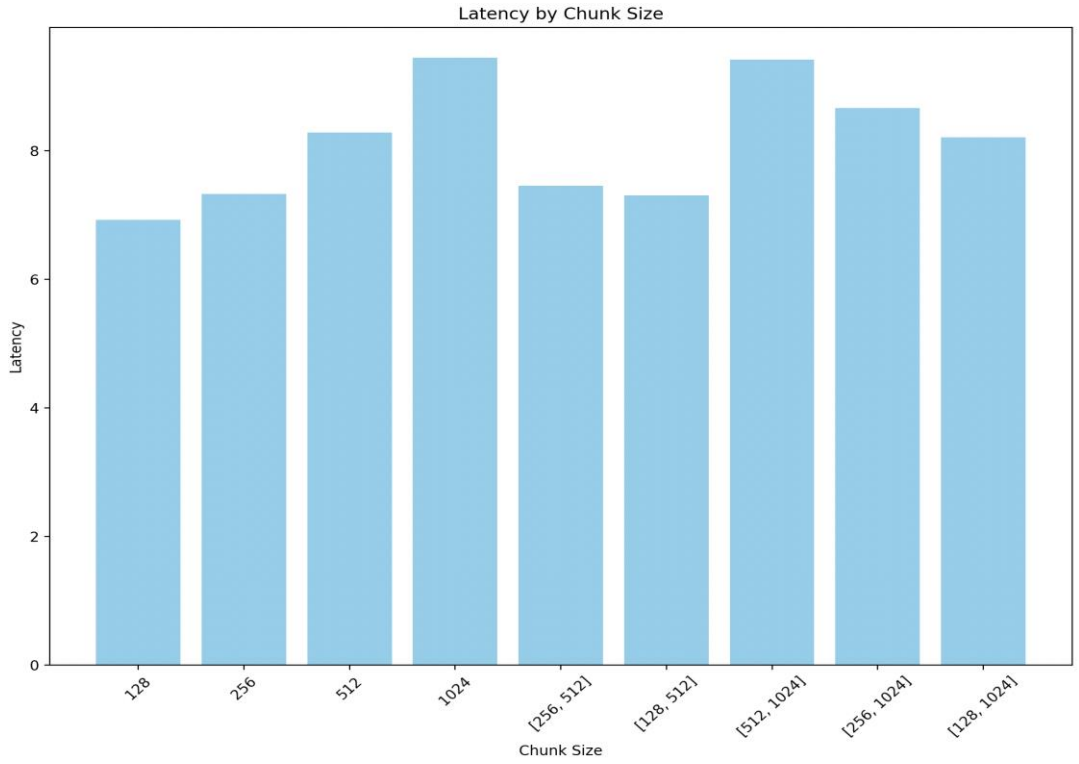
4 장. 연구 결과

4.1 연구 실험 결과

GPT-3.5-turbo-16k 모델의 경우, 청크 사이즈가 증가할수록 Latency 가 상승했다. 128 에서 1024 로 가면서 6.923 초에서 9.441 초로 증가하였는데 이는 처리해야 할 데이터양이 많아지므로 예상되는 결과이다.

GPT-4 모델에서도 유사한 패턴이 나타났지만, 전반적으로 GPT-3.5-turbo-16k 보다 낮은 Latency 값을 기록했다. 특히, 128 청크 사이즈에서 5.532 초로 가장 낮았으며, 1024 에서는 7.599 초로 증가했다.

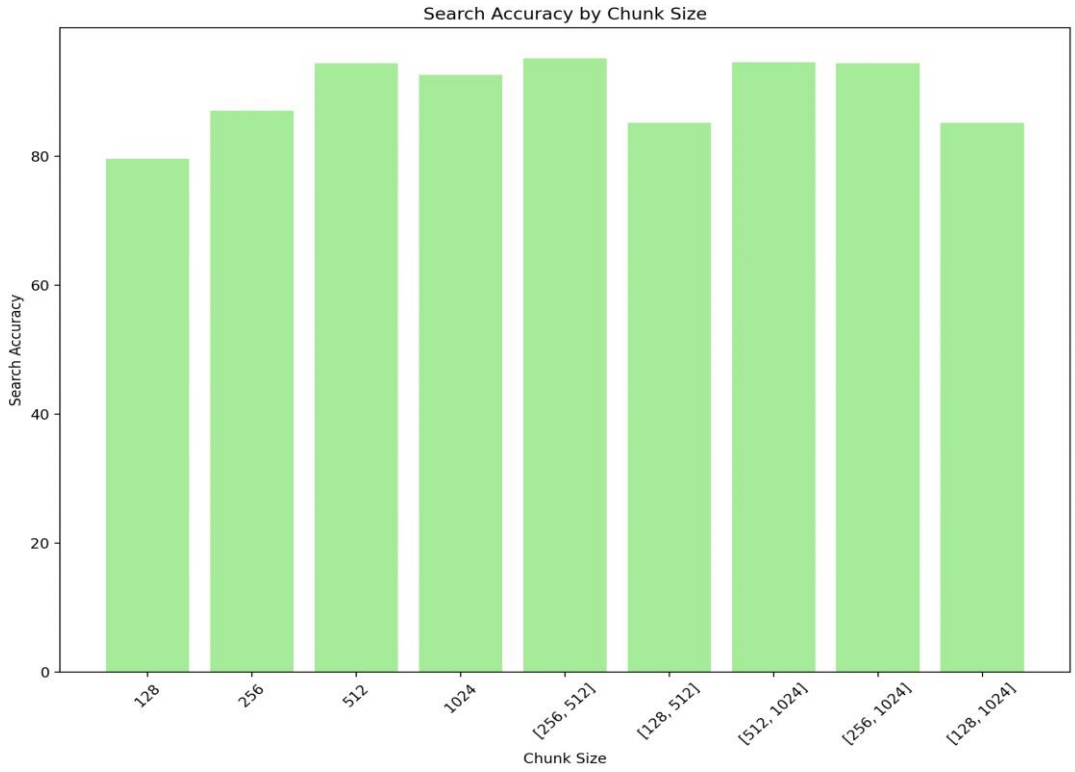
두 청크 사이즈를 조합한 경우 ([128, 1024], [256, 1024], [512, 1024], [128, 512], [256, 512]), Latency 는 단일 청크 사이즈 사용 시보다 다소 높은 경향을 보였다.



[그림 6] 청크 사이즈 별 Latency

검색 정확도는 GPT-3.5-turbo-16k 모델에서 청크 사이즈 1024 에서 92.593 으로 가장 높았으며, GPT-4 모델에서는 512 에서 94.444 로 가장 높은 정확도를 보였다.

두 청크 사이즈를 조합한 경우, GPT-3.5-turbo-16k 모델은 [256, 512]에서 95.146 으로 가장 높은 검색 정확도를 달성했으며, GPT-4 모델은 [256, 1024]에서 94.444 로 높은 정확도를 기록했다.

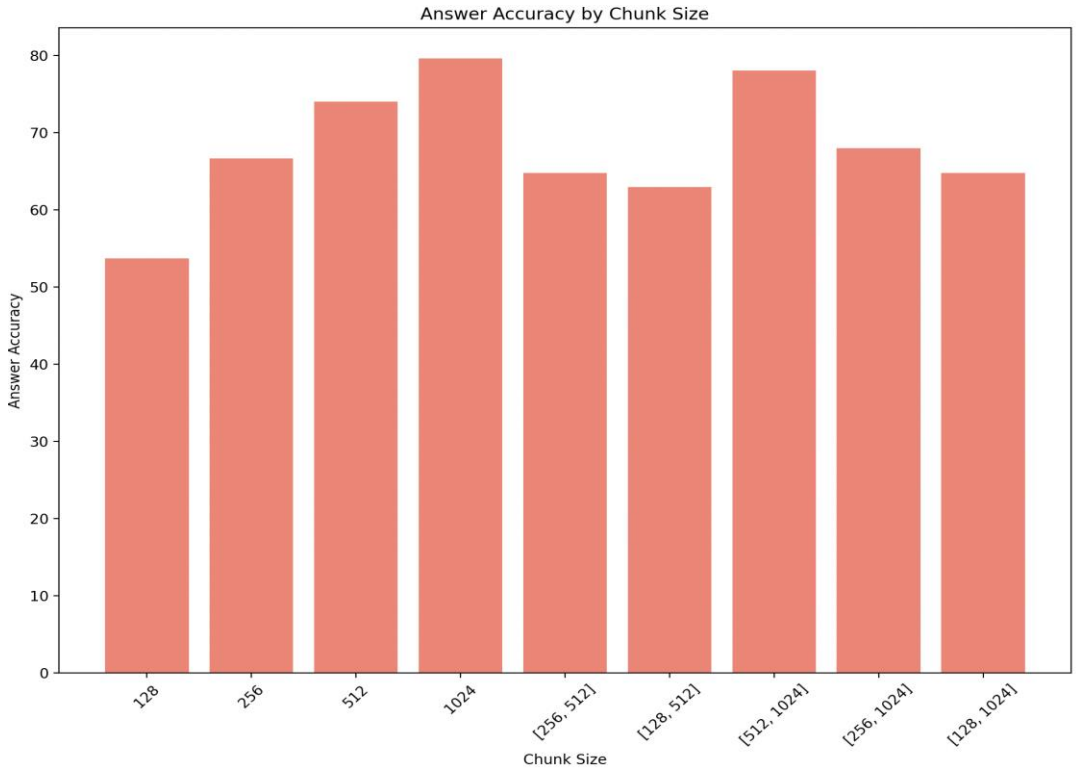


[그림 7] 청크 사이즈 별 검색 정확도

답변 정확도는 GPT-3.5-turbo-16k 모델에서 청크 사이즈 1024 에서 79.630 으로 가장 높았다. 이는 더 큰 청크 사이즈가 더 많은 문맥 정보를 제공하여 더 정확한 답변을 생성할 수 있음을 시사한다.

GPT-4 모델은 청크 사이즈 512 에서 59.259 의 정확도를 보였으나, 청크 사이즈를 조합한 [512, 1024]에서는 69.259 로 상당히 향상되었다.

두 모델 모두 단일 청크 사이즈보다는 조합된 청크 사이즈를 사용했을 때 더 높은 답변 정확도를 보였다. 특히 GPT-3.5-turbo-16k 모델은 [512, 1024] 조합에서 78.049 의 가장 높은 답변 정확도를 기록했다.



[그림 8] 청크 사이즈 별 답변 정확도

실험 결과를 종합해 보면, GPT-3.5-turbo-16k 모델과 GPT-4 모델 간의 성능 차이로 인해 GPT-4 모델은 일반적으로 128 및 256 청크 사이즈에서 더 낮은 Latency 값을 가짐에도 불구하고 512 및 1024 청크 사이즈에서 GPT-3.5-turbo-16k 에 비해 더 높은 검색 정확도와 답변 정확도를 달성했다.

청크 사이즈가 증가함에 따라 Latency 가 증가하는 경향이 있지만, 이는 동시에 검색 정확도와 답변 정확도가 개선되는 것으로 나타났다. 예를 들어, GPT-3.5-turbo-16k 모델을 사용할 때 128 청크 사이즈의 경우 검색 정확도가 79.630 이었지만, 1024 청크 사이즈에서는 92.593 으로 상당히 증가했다.

두 개의 청크 사이즈를 조합한 경우, 단일 청크 사이즈를 사용했을 때보다 더 높은

검색 정확도와 답변 정확도를 달성했다. 특히 [256, 512] 조합은 GPT-3.5-turbo-16k 모델을 사용하여 95.146 의 가장 높은 검색 정확도를 보였다..

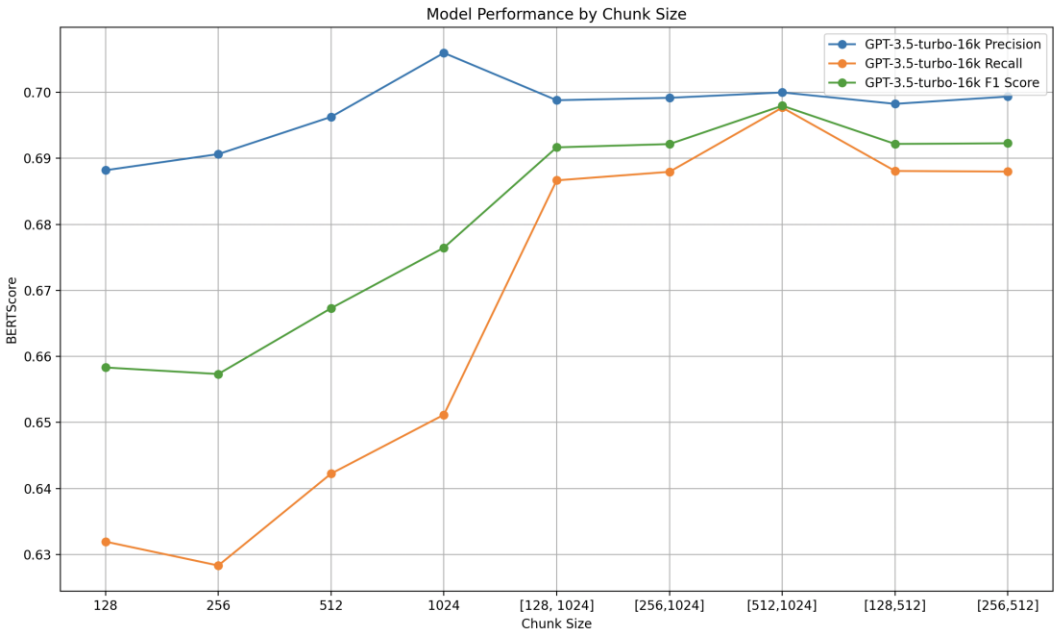
그리고, GPT-4 모델은 [512, 1024] 조합에서 92.859 의 검색 정확도와 69.259 의 답변 정확도로 상대적으로 균형 잡힌 성능을 보였다.

표 6. 실험 결과

청크 사이즈	평가 모델	Latency	검색 정확도	답변 정확도
128	gpt-3.5-turbo-16k	6.923	79.630	51.852
256	gpt-3.5-turbo-16k	7.324	87.037	66.667
512	gpt-3.5-turbo-16k	8.278	91.735	74.074
1024	gpt-3.5-turbo-16k	9.441	92.593	79.630
128	gpt-4	5.532	79.630	53.704
256	gpt-4	5.504	87.037	51.852
512	gpt-4	7.804	94.444	59.259
1024	gpt-4	7.599	90.741	62.963
[128, 1024]	gpt-3.5-turbo-16k	8.205	85.185	64.815
[256, 1024]	gpt-3.5-turbo-16k	8.658	91.236	67.983
[512, 1024]	gpt-3.5-turbo-16k	9.416	94.533	78.049
[128, 1024]	gpt-4	6.064	85.185	55.556
[256, 1024]	gpt-4	6.817	94.444	57.407
[512, 1024]	gpt-4	8.362	92.859	69.259
[128, 512]	gpt-3.5-turbo-16k	7.304	85.185	62.963
[256, 512]	gpt-3.5-turbo-16k	7.454	95.146	64.815
[128, 512]	gpt-4	6.226	85.185	51.852
[256, 512]	gpt-4	7.183	92.593	57.407

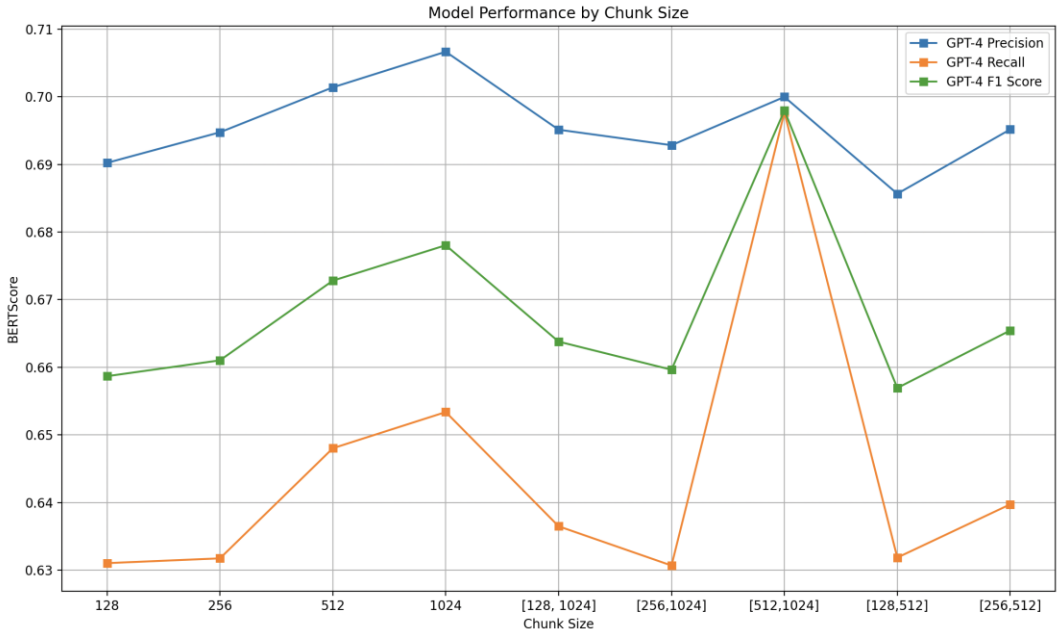
이번에는 GPT-3.5-turbo-16k 와 GPT-4 모델을 사용하여 다양한 청크 크기에 대한 BERTScore 를 측정하였다. 실험 결과, 단일 청크 크기를 사용했을 때보다 두 개의 청크

크기를 혼합하여 사용했을 때의 성능이 전반적으로 우수한 것으로 나타났다. 특히, [512, 1024] 청크 크기 조합을 사용했을 때 GPT-3.5-turbo-16k 모델은 0.699 의 Precision, 0.697 의 Recall, 그리고 0.697 의 F1 Score 를 달성하여 가장 높은 성능을 보였다.



[그림 9] Chunk Size 별 BERTScore 결과(GPT-3.5-turbo-16k)

GPT-4 모델에서도 유사한 경향이 관찰되었다. [512, 1024] 청크 크기 조합은 0.699 의 Precision, 0.697 의 Recall, 그리고 0.697 의 F1 Score 를 기록하며, 단일 청크 크기를 사용했을 때보다 더 높은 성능을 나타냈다.



[그림 10] Chunk Size 별 BERTScore 결과(GPT-4)

이는 다양한 크기의 청크를 혼합함으로써 모델이 더 다양한 맥락에서 정보를 학습하고, 이를 통해 더 정확한 임베딩을 생성할 수 있기 때문으로 해석될 수 있다.

표 7. BERTScore 결과

청크 사이즈	평가 모델	Precision	Recall	F1 Score
128	gpt-3.5-turbo-16k	0.688	0.632	0.658
256	gpt-3.5-turbo-16k	0.691	0.628	0.657
512	gpt-3.5-turbo-16k	0.696	0.642	0.667
1024	gpt-3.5-turbo-16k	0.706	0.651	0.676
128	gpt-4	0.690	0.631	0.659
256	gpt-4	0.695	0.632	0.661
512	gpt-4	0.701	0.648	0.673
1024	gpt-4	0.707	0.653	0.678
[128, 1024]	gpt-3.5-turbo-16k	0.699	0.687	0.692

[256, 1024]	gpt-3.5-turbo-16k	0.699	0.688	0.692
[512, 1024]	gpt-3.5-turbo-16k	0.706	0.651	0.676
[128, 1024]	gpt-4	0.695	0.636	0.664
[256, 1024]	gpt-4	0.693	0.631	0.660
[512, 1024]	gpt-4	0.700	0.698	0.698
[128, 512]	gpt-3.5-turbo-16k	0.698	0.688	0.692
[256, 512]	gpt-3.5-turbo-16k	0.699	0.688	0.692
[128, 512]	gpt-4	0.686	0.632	0.657
[256, 512]	gpt-4	0.695	0.640	0.665

5 장. 결론

이번 연구 결과는 체계적인 실험을 통해 문서 임베딩에 있어서 청킹 전략이 중요한 영향을 미칠 수 있음을 보여준다.

단일 청크 사이즈보다 조합된 청크 사이즈를 사용하는 것이 일반적으로 검색 및 답변 성능 향상에 기여할 수 있음을 확인하였다. 그러나 이러한 향상은 처리 시간 증가와 교환해야 하는 경우가 많으므로, 최적의 성능과 효율성을 달성하기 위해서는 이 두 가지 요소 사이에서 적절한 균형을 찾아야 한다.

이 결과는 도메인 지식을 활용하여 RAG 방식의 LLM 서비스 품질을 향상하는 데 유용한 통찰을 제공할 것이다.

참고문헌

[1] Dale, Robert. "GPT-3: What's it good for?." Natural Language Engineering 27.1 (2021): 113-118.

[2] Vaswani, Ashish, et al. "Attention is all you need." Advances in neural information processing systems 30 (2017).

[3] Lewis, Patrick, et al. "Retrieval-augmented generation for knowledge-intensive nlp tasks." *Advances in Neural Information Processing Systems* 33 (2020): 9459-9474.

[4] Soo-Hwan Lee, & Ki-Sang Song (2023). Prompt engineering to improve the performance of teaching and learning materials Recommendation of Generative Artificial Intelligence. Journal of the Korea Society of Computer and Information , 28(8), 195-204.

[5] Zhao, W. X., Zhou, K., Li,)., Tang, T., Wang, X., Hou, Y., ... & Wen, J. R. (2023). A survey of large language models. arXiv preprint arXIv:2303.18223.

[6] Sutskever, I., Vinyals, O., & Le, Q. V. (2014). Sequence to sequence learning with neural networks. Advances in neural information processing systems, 27.

[7] Bahdanau, D., Cho, K., & Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. arXiv preprint arXiv:1409.0473.

[8] Radford, A., Narasimhan, K., Salimans, T., & Sutskever, I. (2018). Improving language understanding by generative pre-training.

[9] Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., & Sutskever, I. (2019). Language models are unsupervised multitask learners.OpenAl blog, 1(8), 9.

[10] Schulman,j., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017). Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347.

[11] Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., ... & Amodei,

D. (2020). Language models are few-shot learners. Advances in neural information processing systems, 33, 1877-1901.

[12] https://python.langchain.com/docs/modules/data_connection/vectorstores/

[13] https://medium.com/@ktwan6675/guide-for-llm-rag-for-your-own-data-ch-1-kor-49528f4b56ae