SK네트웍스 Family AI과정 3기

모델링 및 평가 테스트 계획 및 결과 보고서

□ 개요

- 산출물 단계 : 모델링 및 평가
- 평가 산출물 : 테스트 계획 및 결과 보고서
- 제출 일자 : 2024-12-26
- 깃허브 경로 : https://github.com/SKNETWORKS-FAMILY-AICAMP/SKN03-FINAL-5Team
- 작성 팀원 : 김종식, 하은진

개요	• 모의면접 질문생성 및 평가 모델 본 결과 보고서는 LLM을 활용한 질문 생성 모델을 구현하고, 해당 모델의 테스트 계획 및 결과를 표시함에 목적이 있다. 이 프로젝트는 사용자의 이력서, 희망 직무 등의 정보를 기반으로 맞춤형 인터뷰 질문을 생성하고, 질문생성 정보를 Vector DB에 저장 및 검색하는 기능을 포함한다.
테스트 계획	• 테스트 목적 이력서 키워드 추출: S3에서 이력서 파일을 로드한 후 주요 키워드를 정상적으로 추출하는지 확인. 맞춤형 인터뷰 생성: 사용자 정보(키워드, 희망 직무, 이력서)를 기반으로 인터뷰 및 인터뷰 ID 생성 여부 확인. Vector DB 검색: Retriever와 reranker를 통한 데이터 검색과 문서 순회 기능 확인. 질문 생성: 사용자 정보와 Vector DB를 기반으로 질문 및 모범답안을 생성하고, 번역 기능의 정상 작동 여부를 확인. 정보 저장: 생성된 질문과 사용자 정보를 DB에 정상적으로 저장하는지 확인.

테스트 과정	1. 이력서 키워드 추출 ● 입력 데이터: S3에 저장된 사용자 이력서 파일(PDF 형식). ● 과정: ○ S3에서 이력서 파일 로드. ○ 키워드 추출 함수 실행. ● 확인사항: ○ 주요 키워드가 이력서 내용에서 정상적으로 추출되었는지 검증. 2. 맞춤형 인터뷰 생성 ● 입력 데이터: 사용자 정보(키워드, 희망 직무, 이력서). ● 과정: ○ 사용자 정보를 입력하여 인터뷰 생성 함수 실행. ○ 인터뷰 ID 자동 생성 확인. ● 확인사항: ○ 인터뷰 및 인터뷰 ID 생성 여부. 3. Vector DB 검색 ● 입력 데이터: 질문과 관련된 문서 데이터. ● 과정: ○ Retriever 검색기 테스트. ○ Reranker를 통한 문서 우선순위 설정 및 정렬 테스트. ● 확인사항: ○ 검색된 문서가 관련성이 높은 순으로 반환되는지 확인. ○ 문서 순회 시 데이터 누락이 없는지 검증. 4. 질문생성 ● 입력 데이터: 사용자 정보(키워드, 이력서)와 Vector DB 데이터. ● 과정: ○ 사용자 정보를 기반으로 질문 생성. ○ 키워드 미존재 시 기본 질문 출력 확인. ○ 질문 및 모범답안 생성 후 영어 번역 확인. ○ 다중 문서에서 질문 생성 시 다양성 확보 여부 확인. ● 확인사항: ○ 질문 및 모범답안의 품질과 관련 데이터의 번역 정확도. 5. 정보저장 ● 입력 데이터: 생성된 질문 및 사용자 정보. ● 과정: ○ 질문과 사용자 정보를 DB 저장 함수에 전달. ○ 저장된 데이터의 정합성 검증. ● 확인사항: ○ 모든 데이터가 DB에 정상적으로 저장되었는지 확인.

결과	• 결과 시스템의 주요 기능(키워드 추출, 인터뷰 생성, Vector DB 검색, 질문 생성, 데이터 저장)이 모두 정상적으로 작동함이 확인 되었다. 

job_question_kor	selected_keyword	job_question_eng	job_solution_kor	job_solution_eng	job_context
Docker를 사용하여 애플리케이션을 컨테이너화할 때 고려해야 할 주요 요소는 무엇이며, 이를 통해 어떤 문제를 해결했는지 설명해 주세요.	Dbms, python, Docker, Azure Cognitive Services, PyTorch, FastAPI, REST API, PostgreSQL, Redis, RNNs, Stable Diffusion, Nodejs, AutoML, Firebase, Hugging Face, React Context API, JS, GAN, React, TensorFlow	Question: What are the key factors to consider when containerizing an application using Docker, and what problems have been solved through this process?	Docker를 사용하여 애플리케이션을 컨테이너화할 때, 핵심 목표는 애플리케이션의 이식성과 배포 효율성을 향상시키는 것이었습니다. 하지만, 다양한 환경에서의 일관성 유지를 보장하는 것이 큰 도전과제로 작용했습니다. 이를 극복하기 위해 Docker파일을 작성하고, CI/CD 파이프라인에 통합하여 자동화된 빌드를 구현했습니다. 그 결과, 배포 시간이 50% 단축되었고, 환경 간의 불일치 문제를 최소화하여 안정성을 크게 향상시켰습니다. 이러한 경험은 앞으로의 프로젝트에서도 컨테이너화를 통해 효율성을 극대화할 수 있는 기반이 될 것입니다.	When containerizing an application using Docker, the core objective was to enhance the application's portability and deployment efficiency. However, ensuring consistent performance across diverse environments posed a significant challenge. To address this, we created a Dockerfile and integrated it into a CI/CD pipeline to implement automated builds. As a result, deployment times were reduced by 50%, and we minimized issues related to environmental discrepancies, significantly improving stability. This experience will serve as a foundation for maximizing efficiency through containerization in future projects.	Skill: Docker Question: 1. Conclusion Answer: DevOps technologies are growing at an exponential pace. As the systems are being more and more distributed, developers have turned towards containerization because of the need to develop software faster and maintain them better. They also aid in easier and faster continuous integration and deployment process which is why these technologies have experienced tremendous growth. Docker is the most famous and popular tool for achieving the purpose of containerization and continuous integration/development and also for continuous deployment due to its great support for pipelines. With the growing ecosystem, docker has proven itself to be useful to operate on multiple use cases thereby making it all the more exciting to learn it!To build a good Dockerfile: https://docs.docker.com/engine/reference/builder/ source: None
Docker를 사용하여 애플리케이션을 컨테이너화하는 과정에서 주요 도전 과제는 무엇이었으며, 이를 해결하기 위해 어떤 접근 방식을 취하였는지 설명해 주세요.	Dbms, python, Docker, Azure Cognitive Services, PyTorch, FastAPI, REST API, PostgreSQL, Redis, RNNs, Stable Diffusion, Nodejs, AutoML, Firebase, Hugging Face, React Context API, JS, GAN, React, TensorFlow	Question: What were the main challenges you faced while containerizing applications using Docker, and what approaches did you take to solve them?	애플리케이션을 컨테이너화하는 과정에서의 핵심 목표는 배포의 일관성과 속도를 향상시키는 것이었습니다. 하지만 추가 설정에서 의존성 문제와 네트워크 구성의 복잡성이라는 도전에 직면했었습니다. 이를 해결하기 위해 Docker Compose를 활용하여 여러 서비스를 정의하고 관리하는 전략을 세웠고, 각 컨테이너의 환경을 명확히 설정하여 문제를 최소화했습니다. 이 과정을 통해 배포 시간을 30% 단축시키고, 애플리케이션의 안정성을 크게 개선할 수 있었습니다. 앞으로 이러한 경험을 바탕으로 더 복잡한 마이크로서비스 아키텍처를 구현하여, 지속적인 통합 및 배포 프로세스를 더욱 효율적으로 발전시킬 수 있을 것입니다.	containerizing applications was to improve the consistency and speed of deployment. However, I faced challenges related to dependency issues and the complexity of network configuration during the initial setup. To address these, I utilized Docker Compose to define and manage multiple services, clearly setting up the environment for each container to minimize problems. Through this process, I was able to reduce deployment time by 30% and significantly improve the stability of the application. Moving forward, based on this experience, I will implement more complex microservices architectures and further enhance the efficiency of continuous integration and deployment.	Skill: Docker Question: 1. Conclusion Answer: DevOps technologies are growing at an exponential pace. As the systems are being more and more distributed, developers have turned towards containerization because of the need to develop software faster and maintain them better. They also aid in easier and faster continuous integration and deployment process which is why these technologies have experienced tremendous growth. Docker is the most famous and popular tool for achieving the purpose of containerization and continuous integration/development and also for continuous deployment due to its great support for pipelines. With the growing ecosystem, docker has proven itself to be useful to operate on multiple use cases thereby making it all the more exciting to learn it!To build a good Dockerfile: https://docs.docker.com/engine/reference/builder/ source: None
Docker를 사용한 프로젝트 경험이 있으시다면, 이를 통해 어떤 문제를 해결했는지 구체적으로 설명해 주세요.	Dbms, python, Docker, Azure Cognitive Services, PyTorch, FastAPI, REST API, PostgreSQL, Redis, RNNs, Stable Diffusion, Nodejs, AutoML, Firebase, Hugging Face, React Context API, JS, GAN, React, TensorFlow	Question: If you have experience with projects using Docker, please explain in detail what problems you solved through it.	프로젝트에서 Docker를 활용하여 애플리케이션의 배포 속도를 50% 향상시키는 것을 목표로 했습니다. 초기에는 다양한 환경에서의 호환성 문제와 설정의 복잡성이라는 장애물에 직면했으나, Docker를 사용하여 컨테이너화된 프로젝 일관된 환경을 제공하고 배포 프로세스를 자동화할 수 있었습니다. 이를 위해 Dockerfile을 작성하라고 CI/CD 파이프라인에 통합하여 코드 변경 시 자동으로 테스트와 배포가 이루어지도록 했습니다. 결과적으로, 배포 시간이 크게 단축되었고, 팀의 생산성이 증가했으며, 향후 프로젝트에서도 Docker를 활용하여 더 효율적이고 유연한 개발 환경을 구축할 수 있음을 확신하게 되었습니다.	With Docker, we aimed to increase the deployment speed of the application by 50%. However, we encountered challenges such as compatibility issues across different environments and the complexity of configurations. However, by utilizing Docker for containerization, we were able to provide a consistent environment and automate the deployment process. We optimized the Dockerfile and integrated it into the CI/CD pipeline to ensure that testing and deployment would occur automatically with code changes. As a result, deployment times were significantly reduced, team productivity increased, and I became confident that using Docker would allow us to build a more efficient and flexible development	Skill: Docker Question: 1. Conclusion Answer: DevOps technologies are growing at an exponential pace. As the systems are being more and more distributed, developers have turned towards containerization because of the need to develop software faster and maintain them better. They also aid in easier and faster continuous integration and deployment process which is why these technologies have experienced tremendous growth. Docker is the most famous and popular tool for achieving the purpose of containerization and continuous integration/development and also for continuous deployment due to its great support for pipelines. With the growing ecosystem, docker has proven itself to be useful to operate on multiple use cases thereby making it all the more exciting to learn it!To build a good Dockerfile: https://docs.docker.com/engine/reference/builder/ source: None

평가모델

개요	• 사용자 답변 평가 모델 RAG의 평가 프레임워크인 RAGAS를 이용하여 사용자 평가 모델의 정량적 지표를 얻고자 함. 이후 LLM모델을 통해 정성적으로 종합평가를 제공함으로써 사용자에게 모의면접 경험과 정보를 제공한다.
테스트 지표	• RAGAS • heystack with RAGAS
테스트 목적	• 평가수단으로써 타당한 RAGAS 지표를 얻고자 함. • 언어 차이에 따라 값에 변화가 있는지 확인하고자 함.
테스트 과정	• RAGAS 지표 평가 위 질문 모델에서 생성된 질문, 모범답안, 참조 context 내용, 임의로 생성한 사용자 답변, 총 4가지로 구성된 데이터셋을 바탕으로 어떤 지표가 일관성 있으며, 실제 텍스트 내용을 설명하기에 타당한지 분석. • 데이터 언어 변환 제공된 데이터셋의 구성을 한국어, 영어로 다양하게 구성해보았을 때 지표의 변화가 있는지 분석.

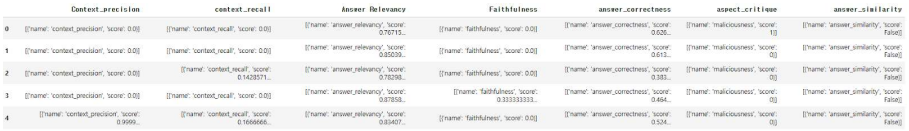
테스트 결과

- RAGAG 지표 평가

- 지표 : context_precision, context_recall, answer_relevancy, faithfulness, answer_correctness, aspect_critique, answer_similarity

- 7가지 지표를 동일한 데이터셋에 도입해보았을 때
    1. 일관성 있는 지표 산출
    2. 정확도를 가진 지표
    3. 사용자 답변 평가로써의 타당성

  을 가진 지표는 answer relevancy, answer correctness로 판단.



	Context_precision	context_recall	Answer Relevancy	Faithfulness	answer_correctness	aspect_critique	answer_similarity
0	[{'name': 'context_precision', 'score': 0.0}]	[{'name': 'context_recall', 'score': 0.0}]	[{'name': 'answer_relevancy', 'score': 0.76715...	[{'name': 'faithfulness', 'score': 0.0}]	[{'name': 'answer_correctness', 'score': 0.626...	[{'name': 'maliciousness', 'score': 1.0}]	[{'name': 'answer_similarity', 'score': False}]
1	[{'name': 'context_precision', 'score': 0.0}]	[{'name': 'context_recall', 'score': 0.0}]	[{'name': 'answer_relevancy', 'score': 0.85039...	[{'name': 'faithfulness', 'score': 0.0}]	[{'name': 'answer_correctness', 'score': 0.613...	[{'name': 'maliciousness', 'score': 0.0}]	[{'name': 'answer_similarity', 'score': False}]
2	[{'name': 'context_precision', 'score': 0.0}]	[{'name': 'context_recall', 'score': 0.1428571...	[{'name': 'answer_relevancy', 'score': 0.78298...	[{'name': 'faithfulness', 'score': 0.0}]	[{'name': 'answer_correctness', 'score': 0.383...	[{'name': 'maliciousness', 'score': 0.0}]	[{'name': 'answer_similarity', 'score': False}]
3	[{'name': 'context_precision', 'score': 0.0}]	[{'name': 'context_recall', 'score': 0.0}]	[{'name': 'answer_relevancy', 'score': 0.87858...	[{'name': 'faithfulness', 'score': 0.3333333333...	[{'name': 'answer_correctness', 'score': 0.464...	[{'name': 'maliciousness', 'score': 0.0}]	[{'name': 'answer_similarity', 'score': False}]
4	[{'name': 'context_precision', 'score': 0.9999...	[{'name': 'context_recall', 'score': 0.1666666...	[{'name': 'answer_relevancy', 'score': 0.83407...	[{'name': 'faithfulness', 'score': 0.0}]	[{'name': 'answer_correctness', 'score': 0.524...	[{'name': 'maliciousness', 'score': 0.0}]	[{'name': 'answer_similarity', 'score': False}]

- 데이터셋 언어 변환

  – 한국어

	Context_precision	context_recall	Answer Relevancy	Faithfulness	answer_correctness
0	[{'name': 'context_precision', 'score': 0.9999...	[{'name': 'context_recall', 'score': 0.0}]	[{'name': 'answer_relevancy', 'score': 0.0}]	[{'name': 'faithfulness', 'score': 0.0}]	[{'name': 'answer_correctness', 'score': 0.319...
1	[{'name': 'context_precision', 'score': 0.0}]	[{'name': 'context_recall', 'score': 0.0}]	[{'name': 'answer_relevancy', 'score': 0.83765...	[{'name': 'faithfulness', 'score': 0.0}]	[{'name': 'answer_correctness', 'score': 0.374...
2	[{'name': 'context_precision', 'score': 0.9999...	[{'name': 'context_recall', 'score': 0.1428571...	[{'name': 'answer_relevancy', 'score': 0.0}]	[{'name': 'faithfulness', 'score': 0.5}]	[{'name': 'answer_correctness', 'score': 0.307...
3	[{'name': 'context_precision', 'score': 0.9999...	[{'name': 'context_recall', 'score': 0.2}]	[{'name': 'answer_relevancy', 'score': 0.0}]	[{'name': 'faithfulness', 'score': 0.25}]	[{'name': 'answer_correctness', 'score': 0.415...
4	[{'name': 'context_precision', 'score': 0.9999...	[{'name': 'context_recall', 'score': 0.1666666...	[{'name': 'answer_relevancy', 'score': 0.0}]	[{'name': 'faithfulness', 'score': 0.0}]	[{'name': 'answer_correctness', 'score': 0.347...

  – context만 영어

	Context_precision	context_recall	Answer Relevancy	Faithfulness
0	[{'name': 'context_precision', 'score': 0.9999...	[{'name': 'context_recall', 'score': 0.0}]	[{'name': 'answer_relevancy', 'score': 0.0}]	[{'name': 'faithfulness', 'score': 0.0}]
1	[{'name': 'context_precision', 'score': 0.9999...	[{'name': 'context_recall', 'score': 0.0}]	[{'name': 'answer_relevancy', 'score': 0.82105...	[{'name': 'faithfulness', 'score': 0.0}]
2	[{'name': 'context_precision', 'score': 0.0}]	[{'name': 'context_recall', 'score': 0.0}]	[{'name': 'answer_relevancy', 'score': 0.0}]	[{'name': 'faithfulness', 'score': 0.0}]
3	[{'name': 'context_precision', 'score': 0.0}]	[{'name': 'context_recall', 'score': 0.0}]	[{'name': 'answer_relevancy', 'score': 0.79909...	[{'name': 'faithfulness', 'score': 0.0}]
4	[{'name': 'context_precision', 'score': 0.0}]	[{'name': 'context_recall', 'score': 0.0}]	[{'name': 'answer_relevancy', 'score': 0.0}]	[{'name': 'faithfulness', 'score': 0.0}]

  – 영어

	Context_precision	context_recall	Answer Relevancy	Faithfulness	answer_correctness
0	[{'name': 'context_precision', 'score': 0.0}]	[{'name': 'context_recall', 'score': 0.0}]	[{'name': 'answer_relevancy', 'score': 0.76718...	[{'name': 'faithfulness', 'score': 0.0}]	[{'name': 'answer_correctness', 'score': 0.626...
1	[{'name': 'context_precision', 'score': 0.0}]	[{'name': 'context_recall', 'score': 0.0}]	[{'name': 'answer_relevancy', 'score': 0.84344...	[{'name': 'faithfulness', 'score': 0.0}]	[{'name': 'answer_correctness', 'score': 0.592...
2	[{'name': 'context_precision', 'score': 0.0}]	[{'name': 'context_recall', 'score': 0.1428571...	[{'name': 'answer_relevancy', 'score': 0.78293...	[{'name': 'faithfulness', 'score': 0.0}]	[{'name': 'answer_correctness', 'score': 0.383...
3	[{'name': 'context_precision', 'score': 0.9999...	[{'name': 'context_recall', 'score': 0.0}]	[{'name': 'answer_relevancy', 'score': 0.87858...	[{'name': 'faithfulness', 'score': 0.333333333...	[{'name': 'answer_correctness', 'score': 0.464...
4	[{'name': 'context_precision', 'score': 0.9999...	[{'name': 'context_recall', 'score': 0.1666666...	[{'name': 'answer_relevancy', 'score': 0.83406...	[{'name': 'faithfulness', 'score': 0.0}]	[{'name': 'answer_correctness', 'score': 0.524...

  – 언어 변환 시 같은 내용임에도 불구하고 지표 변화가 존재하였음.

  – 자체로 평가했던 결과와 가장 유사하게 지표를 산출하는 '영어' 데이터셋으로 평가를 진행하고자 함.