BCE 패턴 기반 마이크로서비스 아키텍처의 배치 유형 식별을 위한 생성형 AI 파인튜닝 방법

조대영[1O], 정수민[2], 박준석[3], 염근혁[4*교신저자]
[1]부산대학교 정보융합공학과 석사과정
[2]부산대학교 정보융합공학과 박사과정
[3]부산대학교 지능물류빅데이터연구소 연구교수
[4*]부산대학교 정보컴퓨터공학부 교수
{jdy08, sumin2708, pjs50, yeom}@pusan.ac.kr

Generative AI Fine-tuning Method for Identifying Deployment Types of BCE Pattern-based Microservice Architecture

Daeyeong Cho[1O], Sumin Jeong[1], Joonseok Park[2], Keunhyuk[3*]
[1]Dept. of Information Convergence Engineering, Pusan National University
[2]Research Institute of Intelligent Logistics Big Data, Pusan National University
[3*]School of Computer Science and Engineering, Pusan National University

요 약

마이크로서비스 아키텍처는 마이크로서비스 간 약결합을 통한 높은 확장성과, 개별 배포를 통한 유지보수성을 제공하는 애플리케이션 구축 방법이다. 그러나, 마이크로서비스 아키텍처는 표준적인 배치방식이나 연결 방법이 부족하여, 마이크로서비스 아키텍처의 전문적인 지식 없이 마이크로서비스 단위를 구분하고 약결합 구조를 배치하기에는 어려움이 있다. 따라서, 본 논문에서는 마이크로서비스 아키텍처의 BCE 패턴 기반 배치 방안으로 마이크로서비스의 기능 및 약결합 구조를 생성형 AI로 학습하는 방법을 제시한다. 제안하는 방법에 따라 생성형 AI 모델인 GPT-3.5-turbo를 바탕으로 파인튜닝 한 결과 파인튜닝 모델을 활용한 배치 정답률이 14% 향상되는 것을 확인하였다. 또한, 파인튜닝 학습 요소의 반영률을 조절하여 모델의 비교 평가를 수행한 결과로 f1-score가 0.019 증가한 것을 통해 파인튜닝 요소가 정답을 결정하는 데 필요한 요소임을 확인하였다.

1. 서론

최근 IT 기업들은 애플리케이션 규모의 증가에 따라 확장성이 높고 독립적인 배포가 가능한 단위인 마이크로서비스의 약결합으로 애플리케이션을 구축하는 마이크로서비스 아키텍처(Microservice Architecture)[1]를 도입하고 있다.

그러나, 마이크로서비스 아키텍처를 특정 설계 패턴에 기반하지 않고 구축하는 경우 마이크로서비스의 크기와 결합 방식이 설정되지 않아 개별 마이크로서비스의 구분이 모호해지고 서비스 간 결합도가 높아진다. 즉, 마이크로서비스 아키텍처를 적용하기 위한 접근성과 유지보수성이 낮은 시스템이 구축될 수 있다. 따라서, 본 논문에서는 소프트웨어 디자인 패턴 중 하나인 BCE(Boundary - Control - Entity)패턴을 활용하여 마이크로서비스를 기능 단위로 분리하고 약결합한다.

하지만, BCE 패턴이 적용된 마이크로서비스 아키텍처를 구축하기 위해서는 마이크로서비스 개발자의 인사이트가 필요하며, 마이크로서비스의 유형을 판단하고 배치하는데 경험에 의존하는 수동적 메커니즘을 활용하고 있다. 따라서, BCE 패턴을 활용하여 마이크로서비스 배치 및 결합을 지원하는 방안으로 대화 기반으로 배치를 결정할 수 있는 생성형 AI 기반의 배치 기법을 제안한다.

2. 관련 연구

A. S. Alsayed 등[2]은 사용자의 자연어 질의에 기반한 마이크로서비스를 추천하는 생성형 AI 학습 방법을 제시하였다. 해당 연구에서는 축적된 사용자 질문과 마이크로서비스 관련 문서를 의미적으로 연계한 학습을 수행하였다. 반면, 본 연구는 BCE 패턴 기반의 마이크로서비스 배치를 지원하는 방안으로 약결합과 기능 명세를 학습한다는 점에서 차이점이 있다.

김시현 등[3]은 모놀리식에서 마이크로서비스를 식별하는 기존 기법을 개선하는 마이크로서비스 식

별 방법을 제안하였다. 해당 연구는 TF-IDF(Term Frequency-Inverse Document Frequency) 가중치를 도출하고, KNN(K-Nearest Neighbor) 알고리즘을 통해 마이크로서비스를 분류하고 식별한다. 본 논문에서는 BCE 패턴 기반의 마이크로서비스를 생성형 AI를 활용하여 식별한다는 점에서 차이점이 있다.

3. BCE 패턴 기반 마이크로서비스를 식별하는 생성형 AI 학습 기법

3.1 마이크로서비스 명세 도출

마이크로서비스 명세는 마이크로서비스의 연결 관계를 나타내는 약결합 명세와 마이크로서비스 기능을 설명하는 기능 명세로 구분된다.

<표 1>는 이전 연구[4]에서 도출한 약결합 명세 요소를 나타낸 것이다.

<표 1> 약결합 명세 요소

리소스	구성요소	하부 구성요소	설명
Pod	name		구분자
	label		약결합 대상 구분자
	port		파드 측면 약결합 연결 요소
	env		컨테이너 환경 변수
	image		컨테이너 기능
Service	name		구분자
	port		서비스 측면 약결합 연결 요소
	type		네트워크 형태 구분
	targetPort		파드 연결 시 포워딩 규칙
	label	selector	파드 연계 정보
Deployment	name		구분자
	template	label	파드 명세
		port	
		env	
		image	

<표 1>의 약결합 명세 요소는 마이크로서비스 간 약결합 수행을 위한 컨테이너(쿠버네티스 리소스)를 기준으로 도출하였다. 파드(Pod)는 컨테이너를 구분하는 리소스로 마이크로서비스가 배치되는 공간이다. 서비스(Service)는 컨테이너 간 약결합 대상이 되는 인터페이스 역할을 수행한다. 디플로이먼트(Deployment)는 파드를 패키징하고 소속된 파드의 관리를 수행한다.

기능 명세는 개별 마이크로서비스의 기능과 역할을 설명하기 위한 요소들을 나타낸 것이다. <표 2>는 도출한 기능 명세의 구성요소를 나타낸 것이다.

<표 2> 기능 명세 구성요소

구성요소		설명
서비스	이름	마이크로서비스를 구분
	설명	마이크로서비스의 기능 및 역할 설명
기반 이미지	이름	마이크로서비스가 사용하는 기반 이미지를 구분
	설명	마이크로서비스가 사용하는 기반 이미지의 기능 설명

<표 2>에서 서비스는 애플리케이션 상에서 각 마이크로서비스 측면의 기능 설명이다. 기반 이미지는 각 마이크로서비스가 사용하는 이미지에 대한 설명이다.

3.2 마이크로서비스 명세 기반 학습 방법

생성형 AI의 학습은 지도학습 방법에 기반하므로, 도출한 명세의 BCE 패턴 답안을 생성하는 과정이 필요하다. <표 3>는 마이크로서비스의 BCE 패턴 답안을 도출하는 기준을 나타낸다.

<표 3> BCE 패턴 마이크로서비스 답안 기준

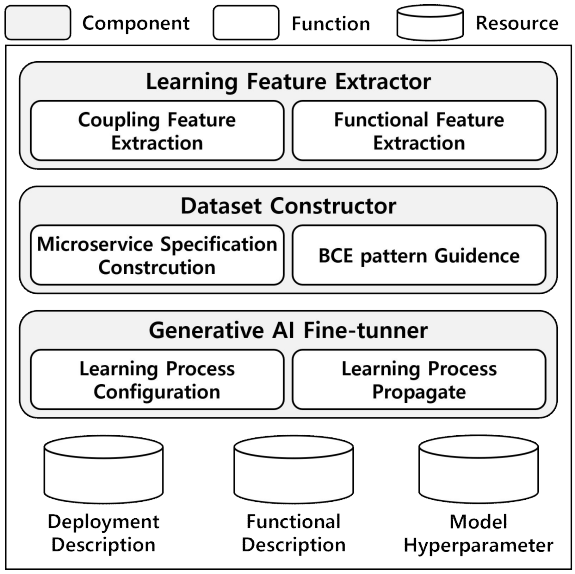
특징 구분	BCE 패턴	설명
약결합 구조적 특징	Boundary	• 시스템의 외부로부터 요청을 전달받으며, 이를 위해 외부와 통신하기 위한 포트가 개방되어 있음 • 전달받은 요청을 Control 마이크로서비스로 전달함
	Control	• 시스템 내부의 Boundary 또는 다른 Control 마이크로서비스로부터 요청을 전달받음 • 전달받은 요청을 다른 Control 또는 Entity 마이크로서비스로 전달하거나 반환만 함
	Entity	• 시스템 내부의 Control 마이크로서비스로부터 요청을 전달받음 • 요청을 다른 마이크로서비스로 전달하지 않고 반환만 함
기능적 특징	Boundary	• 시스템 외부와 소통 • 시스템 외부에 뷰를 제공
	Control	• 특정 비즈니스를 처리 • 데이터를 가공하기 위한 기능 제공
	Entity	• 기능 수행에 필요한 데이터 저장 • 데이터 관리 기능 제공

<표 3>에서 구조적 특징은 마이크로서비스 간의 통신 구조를 나타낸 것으로, 마이크로서비스가 직접적으로 통신하는 요소나 노출되는 범위로 BCE 패턴 유형을 결정한다. 기능적 특징은 마이크로서비스가 수행하는 기능의 특징과 결과 표현 방법에 따라 나

타낸 것으로, 시스템 내에서 마이크로서비스가 수행하는 역할이나 제공하는 기능에 따라 BCE 패턴 유형을 결정한다.

3.3 생성형 AI 학습 아키텍처

BCE 패턴 기반의 마이크로서비스를 생성형 AI에 학습하기 위한 아키텍처는 (그림 1)과 같다.



(그림 1) BCE 패턴 기반 마이크로서비스 학습 구조

(그림 1)의 생성형 AI 학습 아키텍처는 학습 특징 추출 컴포넌트(Learning Feature Extractor), 데이터세트 구축 컴포넌트(Dataset Constructor), 생성형 AI 파인튜닝 컴포넌트(Generative AI Fine-tunner)로 구분된다.

학습 특징 추출 컴포넌트는 BCE 패턴 마이크로서비스에서 <표 1>과 <표 2>에 해당하는 약결합 및 기능 명세를 추출한다. 데이터세트 구축 컴포넌트는 도출된 약결합 및 기능 명세를 <표 3>의 마이크로서비스 특징에 기반하여 생성형 AI 학습을 위한 데이터세트로 구성한다. 생성형 AI 파인튜닝 컴포넌트는 학습 과정을 설정하고, 학습 관련 사항을 생성형 AI에 전달하여 재학습을 수행한다.

4. 실험 및 평가

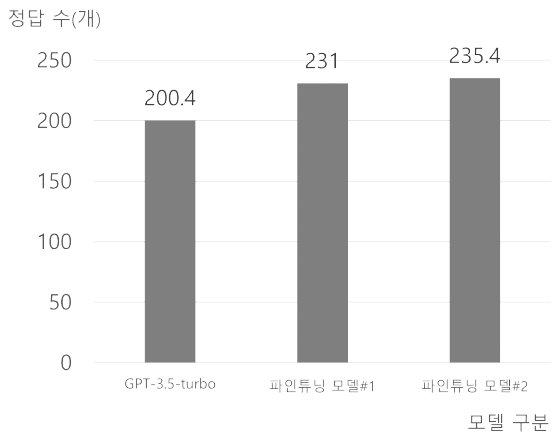
생성형 AI 학습 및 평가를 위해 오픈소스 마이크로서비스 애플리케이션을 깃허브(GitHub)[5] 레포지토리에서 수집하였으며 내역이 <표 4>와 같다.

<표 4> 마이크로서비스 애플리케이션 레포지토리

특징 구분	레포지토리
학습용 애플리케이션 레포지토리	istio/istio
	Microservice-API-Patterns/LakesideMutual
	nginxinc/mra-ingenious
	EdwinVW/pitstop
	lelylan/lelylan
	benwilcock/cqrs-microservice-sampler
	SteeltoeOSS/Samples
	jferrater/Tap-And-Eat-MicroServices
	sambit77/Bookstore
	kbastani/cloud-native-microservice-strangler-example
평가용 애플리케이션 레포지토리	Azure-Samples/aks-store-demo
	GoogleCloudPlatform/microservices-demo
	Staffjoy/v2

본 논문의 실험은 GPT-3.5-turbo(파인튜닝 되지 않은 기반 모델), 파인튜닝 모델#1(기능 명세 기반 파인튜닝 모델), 파인튜닝 모델#2(기능 명세, 약결합 명세 기반 파인튜닝 모델)를 구성하여 수행하였다. 또한, 파인튜닝 모델#1과 파인튜닝 모델#2는 <표 4>의 학습용 애플리케이션 레포지토리를 활용하여 파인튜닝을 수행하였다.

(그림 2)는 <표 4>의 평가용 애플리케이션 레포지토리를 활용하여 BCE 패턴 식별 시 평균 정답 수를 나타낸 것이다. 평가용 애플리케이션 레포지토리의 마이크로서비스의 전체 수는 25개이다. 본 실험의 정확성을 높이기 위하여 파인튜닝 모델 #1과 #2는 각각 파인튜닝을 5회 수행하여 모델을 생성하였다(GPT-3.5-turbo 1개, 파인튜닝 모델 #1 5개, 파인튜닝 모델 #2 5개). 또한, 파인튜닝 모델별로 25개의 마이크로서비스에 대한 BCE 패턴 식별을 10회 반복(총 250회)하여 마이크로서비스 정답률의 평균을 도출하였다.



(그림 2) 모델별 평균 정답 수

실험 결과로 GPT-3.5-turbo의 평균 정답 수는 200.4회, 정답률 80.16%가 도출되었다. 파인튜닝 모델#1의 경우 평균 정답 수는 231회, 정답률 92.4%가 도출되었다. 또한, 파인튜닝 모델#2의 경우 평균 정답 수는 235.4회, 94.16%가 도출되었다. 실험 결과에서 제안하는 방법인 파인튜닝 모델#2는 GPT-3.5-turbo 대비 정답률이 14% 향상된 것을 확인하였다.

<표 1>-<표 3>까지 분석한 요소가 모델에 미치는 영향을 평가하기 위해 파인튜닝 모델#1과 파인튜닝 모델#2를 정밀도(precision), 재현율(recall), f1-score 측면에서 도출한 것이 <표 5>, <표 6>과 같다. 정밀도는 모델 평가 요소 중 모델 예측값이 참인 결과물 중 실제 값도 참인 경우의 비율을 나타낸다. 재현율은 참으로 분류된 실제 값 중 모델 예측값도 참값으로 분류한 경우의 비율을 나타낸다. f1-score는 정밀도와 재현율의 조화 평균을 나타낸다.

<표 5> 파인튜닝 모델#1의 정밀도, 재현율, f1-score

분류 지표 / 클래스 및 평균	precision	recall	f1-score	number of samples
boundary	0.980	0.833	0.901	300
control	0.934	0.940	0.937	750
entity	0.833	1.000	0.909	200
weighted avg	0.928	0.923	0.923	1250

<표 6> 파인튜닝 모델#2의 정밀도, 재현율, f1-score

분류 지표 / 클래스 및 평균	precision	recall	f1-score	number of samples
boundary	0.900	1.000	0.947	300
control	1.000	0.902	0.948	750
entity	0.833	1.000	0.909	200
weighted avg	0.949	0.941	0.942	1250

실험 결과로 파인튜닝 #1보다 파인튜닝 모델#2의 평균 f1-score가 0.019 증가한 것을 확인하였다. 이는, 약결합 명세, 기능 명세를 반영하여 모델의 정답률이 증가된 것을 나타낸다.

5. 결론

본 논문에서는 마이크로서비스를 배치하는 방법으로 BCE 패턴을 적용하였고, 마이크로서비스의 BCE 패턴 유형 식별을 위해 생성형 AI를 학습하는 방법을 제시하였다. 제시하는 방법을 평가하기 위해 GPT-3.5-turbo 모델과 마이크로서비스 명세를 파인튜닝한 모델을 비교하였다. 평가 결과로 파인튜닝을 수행한 모델의 정답률이 기반 모델과 비교하여 14% 향상되었다. 또한, 본 연구에서 제안하는 학습 요소가 파인튜닝 시 정답률에 미치는 영향을 파악하기 위해 수행한 실험에서 제안한 방법의 f1-score가 0.019 높은 것을 확인하였다. 이는, 파인튜닝 시 본 연구에서 제시한 약 결합과 기능 명세 요소들을 반영하는 것이 유의미함을 나타낸다. 향후 본 연구에서 제안한 마이크로서비스 유형 식별 방법을 활용한 마이크로서비스 구축 프로세스를 정립하는 연구를 수행할 것이다.

사사

본 연구성과물은 2024년도 정부(교육부)의 재원으로 한국연구재단의 지원을 받아 수행된 기초연구사업임 (No. RS-2023-00243156)

참고문헌

[1] L. D. S. B. Weerasinghe and I. Perera, “Reference Architecture for Microservices with an Optimized Inter-Service Communication Strategy,” 2024 International Research Conference on Smart Computing and Systems Engineering (SCSE), pp. 1-6, 2024.

[2] A. S. Alsayed, H. K. Dam and C. Nguyen, “MicroRec: Leveraging Large Language Models for Microservice Recommendation,” IEEE/ACM 21st International Conference on Mining Software Repositories, pp. 419-430, 2024.

[3] 김시현, 오재원, “마이크로서비스의 자동 식별을 위한 효과적인 재사용 기반 접근 방법,” 한국정보통신학회논문지, Vol. 27, No. 6, pp. 673-687, 2023.

[4] 조대영, 정수민, 박준석, 염근혁, “생성형 AI를 적용한 컨테이너 기반의 마이크로서비스 배치 기법,” 2024 한국컴퓨터종합학술대회, pp. 313-315, 2024.

[5] GitHub, https://docs.github.com/en/get-started/start-your-journey/about-github-and-git