장예훈 포트폴리오

목차

- **머신러닝 기반 유효특허 분류기 개발에 관한 연구**
  - 개요
  - 이슈 및 해결 방안
  - 결론

- **서버 모니터링 시스템 개발**
  - 개요
  - 이슈 및 해결 방안
  - 결론

- **QoE Data를 이용한 이상감지 탐지**
  - 개요
  - 이슈 및 해결 방안
  - 결론

머신러닝 기반 유효특허 분류기 개발에 관한 연구

머신러닝 기반 유효특허 분류기 개발에 관한 연구 - 개요

목적

특허 데이터와 머신러닝을 활용하여 유효 특허 분류기(Garbage Patent Classifier) 개발

기간

2017.09 - 2018.03 (약 8개월)

배경

- 많은 양의 특허 문서 검토 후 유효 특허 여부 판별 → 막대한 인력, 시간 등의 비용 소모 & 수작업으로 인한 오차, 중복 발생
- 비용과 오차 감소를 위해 유효 특허 분류기 개발

역할

데이터 수집, 분석 및 유효특허 분류기 개발

Data & Tool

- US 특허 데이터
- Pandas, Numpy, Scikit-Learn 등

머신러닝 기반 유효특허 분류기 개발에 관한 연구 – 개요

연구 방법

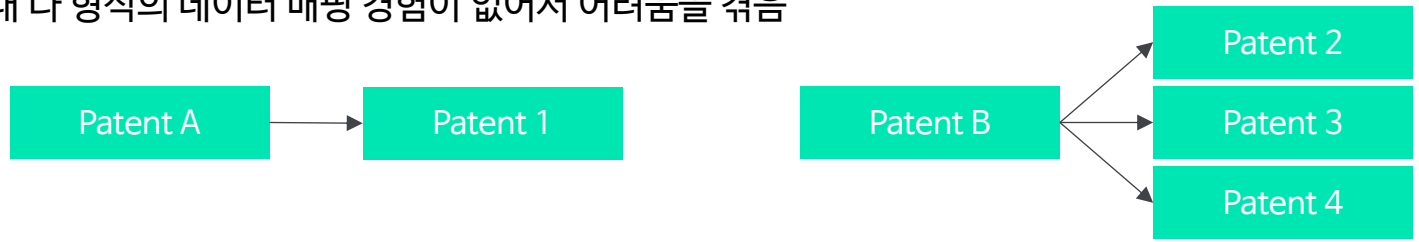
1. 실험 데이터
   - 특허나라와 WIPS DB에서 US 특허 데이터 추출
2. 데이터 전처리
   - 서지적, 네트워크, 텍스트 데이터 가공 후 학습 데이터 생성
3. 실험
   - recall(재현율) 점수를 높이기 위해 다양한 기법 활용

이슈

1. 네트워크 데이터 매핑
2. 성능 저조 1 – 데이터 불균형
3. 성능 저조 2 – 학습 정보 부족

1. 네트워크 데이터 매핑

- **특허의 네트워크 데이터**
  - 유효 특허 간 연관성이 있는 assignee(양수인), application citation(출원 인용), grant citation(등록 인용) 등
  - 하나의 특허에 대해 다수의 네트워크 데이터
  - 1 대 다 형식의 데이터 매핑 경험이 없어서 어려움을 겪음



- **해결 방안**
  - 네트워크 데이터를 별도의 edge list 생성 후 pivot table 로 변환
  - pivot table을 pivot으로 변환 후 학습 데이터 프레임에 병합

Patent A	Patent 1
Patent B	Patent 2
Patent B	Patent 3
Patent B	Patent 4

→

Patent A	Patent 1
Patent B	Patent 2
	Patent 3
	Patent 4

→

	Patent 1	Patent 2	Patent 3	Patent 4
Patent A	1	0	0	0
Patent B	0	1	1	1

2. 성능 저조 1 – 데이터 불균형

- **데이터 불균형**
  - 데이터 탐색 단계에서 EDA로 확인할 수 있는 부분이었으나 EDA 개념과 중요성 인지 부족으로 간과
  - 모델의 accuracy는 높았지만 전부 0 으로 분류하는 현상
  - 전체 데이터 중 극소수의 유효 특허 분류의 성능을 평가하기 위한 지표로 삼은 recall 점수도 저조

	recall	f1-score
0	1.00	0.99
1	0.29	0.40
accuracy	0.99	

Decision Tree

	recall	f1-score
0	1.00	1.00
1	0.57	0.73
accuracy	0.99	

Random Forest Classifier

- **해결 방안**
  - 오버 샘플링 기법 중 소수 클래스 데이터를 보간하여 새로운 데이터를 합성해내는 SMOTE 기법 채택

	recall	f1-score
0	0.99	0.99
1	0.57	0.44
accuracy	0.98	

Decision Tree

	recall	f1-score
0	1.00	1.00
1	0.71	0.83
accuracy	1.00	

Random Forest Classifier

3. 성능 저조 2 – 학습 정보 부족

- **학습 정보 부족**
  - 데이터 불균형 이슈를 해결해도 성능 향상이 만족스럽지 않음
  - 학습 데이터 정보 편향 (수치, 카테고리 데이터만 존재) 가설

	recall	f1-score
0	0.99	0.99
1	0.57	0.44
accuracy	0.98	

Decision Tree

	recall	f1-score
0	1.00	1.00
1	0.71	0.83
accuracy	1.00	

Random Forest Classifier

- **해결 방안**
  - abstract(요약), claim(청구항), title(특허명) 등의 텍스트 데이터 추가
  - 텍스트 데이터의 TF-IDF 값을 추출하여 학습 데이터 프레임에 병합

	recall	f1-score
0	0.96	0.98
1	0.71	0.30
accuracy	0.95	

Decision Tree

	recall	f1-score
0	1.00	1.00
1	0.86	0.92
accuracy	0.99	

Random Forest Classifier

머신러닝 기반 유효특허 분류기 개발에 관한 연구 - 결론

결론

- 네트워크 데이터 추가로 인해 학습 데이터가 sparse 해졌고, 학습 속도가 느려짐 → 차원 축소 등을 활용하여 사이즈 축소 필요
- US 특허 한정 분류 모델 → KR, JP, EU등 여러 국가의 유효 특허 분류 확장 가능

느낀 점

- 학부 수준 작은 규모의 연구였지만 완성하기 위해 노력하고 그 과정에서 책임감과 포기하지 않고 추진하는 힘을 기를 수 있었음
- 데이터 전처리 과정을 경험하며 pandas의 다양한 기법과 활용 방법을 익힐 수 있었고, 대규모의 데이터를 처리할 수 있었던 좋은 경험

아쉬웠던 점

- 당시 pandas 활용능력이 낮아서 데이터 전처리 허점이 많이 발견됐고, python도 보다 효율적으로 구현할 수 있었는데 그러지 못한 아쉬움
- 인터뷰를 준비하면서 기록의 중요성을 느낌 → 텍스트 데이터 추가 작업 코드 부재, 코드의 비체계화, 코드 설명 부족 등

실무 적용 방안

- 실제 데이터를 다룬 경험, 데이터 특성과 모델의 목적을 고려하여 실험을 설계하는 방법
- 데이터 수집, 전처리, 모델 설계 과정에 대한 이해
- 개발도 중요하지만 장기적인 발전을 위해서 내용을 체계적으로 정리하고, 기록하는 습관
- 책임감과 목표 달성 능력 등 논문 작업을 통해 얻은 것을 발판 삼아 모든 일을 적극적으로 임할 수 있는 사람이 되고자 하는 마음가짐을 갖게 됨

서버 모니터링 시스템 개발

서버 모니터링 시스템 개발 - 개요

목적

여러 플랫폼 기반 서버의 효율적인 모니터링을 위한 통합 대시보드 시스템 개발

기간

2018.09 - 2018.12 (약 4개월)

배경

- 서버 모니터링을 위해 당직 근무 시 여러 플랫폼을 모두 확인해야 하는 번거로움 존재
- AWS EC2 (Cloud Watch) 와 Elastic Search 서버의 통합 모니터링 시스템 필요
- Grafana 를 활용하여 서버 모니터링 시스템 구축 요구

역할

AWS EC2 & Elastic Search 서버 Grafana 연동, 대시보드 구성, 매뉴얼 작성, 메신저 Alert 등

Data & Tool

- AWS EC2, Cloud Watch 정보, Elastic Search Log 데이터
- Grafana, Elastic Search, Logstash, Docker 등

서버 모니터링 시스템 개발 - 개요

개발 방법

1. 모니터링 도구 탐색 및 채택 - Grafana
2. 로컬 서버와 Docker 환경 설정 및 Grafana 설치
3. Elastic Search 로그 데이터 샘플과 AWS Cloud Watch 를 Grafana에 연동
4. 대시보드 구성 및 Alert 기능 설정
5. 매뉴얼 작성

이슈

1. 서버, 모니터링 분야 지식 부족
2. Grafana 자료 부족
3. 실제 서버에서 작업 불가

서버 모니터링 시스템 개발 - 이슈

1. 서버, 모니터링 분야 지식 부족

- Docker, AWS EC2, Elastic Search 에 대한 이해와 네트워크, 서버에 대한 지식 부족
- 위 도구들과 서버, 네트워크에 대한 기반 지식 없이 서버 모니터링 시스템 개발 불가

- **해결 방안**
  - Docker, Elastic Search 웨비나, 공식 문서 등을 활용하여 로컬 서버에서 실습하며 지식을 쌓음
  - AWS 가이드를 실습하고, 실험용 인스턴스를 생성하는 등 구조와 원리를 익히려고 노력함
  - Elastic Search 로그 데이터 샘플을 활용하여 Log Stash → Elastic Search 데이터 흐름을 확인하고 익힘

2. Grafana 자료 부족

– 당시 Grafana 튜토리얼이나 예시, 실제 적용 사례 등 관련 자료 부족

– 벤치마크 대상이 없어 시스템 구축 방향 설정에 난항

- **해결 방안**
  – 공식 문서와 Grafana Lab 커뮤니티 활용
  – Grafana 기본 기능, 제공 API 등을 직접 실험하며 기능을 익힘
  – 구체적인 시스템 구축 방향 설정을 위해 주기적인 상부 보고와 피드백으로 시스템 개선

3. 실제 서버에서 작업 불가

–서비스가 운영 중인 서버의 안정성을 위해 실험적인 작업 활동 불가

- **해결 방안**
  – 로컬 서버에 비슷한 환경 구성 후 요구사항 충족을 위한 실험 진행
  – 시스템 완벽 구현 후 모니터링 전용 서버에 시스템 구축과 서비스 서버 연동 진행

서버 모니터링 시스템 개발 - 결론

결론

- 서버 통합 모니터링 시스템 구축으로 모니터링 당직 업무 효율 향상
- 혼자서 실무에 도움이 되는 시스템 전체를 구축한 값진 경험

느낀 점

- 새로운 분야에 겁 먹지 않고 차근차근 처음부터 시작하면 된다는 것을 몸소 깨달음
- 매뉴얼을 제작하고, 지원 요청을 받는 과정에서 시스템 책임자로서 성취감

아쉬웠던 점

- 상부 보고를 위한 문서 외엔 코드나 기타 자료 등을 남겨놓지 않은 것

실무 적용 방안

- ELK 를 실습하며 빅데이터 처리에 대한 이해와 관심
- 공식 문서를 활용하게 되면서 임기응변이 아닌 정석대로 근본적인 문제 해결 가능
- 앞으로 만날 새로운 데이터, 도메인의 기술을 접하고 배우고 익히며 활용하는 데 좀 더 도전적으로 임할 수 있음

QoE 데이터를 이용한 이상징후 감지

QoE 데이터를 이용한 이상징후 감지 - 개요

목적

QoE (Quality of Experience) 데이터를 분석 하여 주기적으로 발생하는 에러 예측 및 신속한 원인 파악

기간

2018.11 - 2019.01 (약 3개월)

배경

- 서버 모니터링 시스템은 사후 대응 체계 / 3교대 모니터링 당직 근무 병행
- 에러 코드와 전조 증상 등의 패턴을 파악하여 실시간으로 이상징후를 감지하는 모델로 사전 예방

역할

QoE 데이터 전처리 일부 및 Resampling, SMOTE 를 활용하여 모델 학습

Data & Tool

- Dtv 의 QoE 데이터
- AWS S3, Pandas, Numpy, Scikit-Learn 등

QoE 데이터를 이용한 이상징후 감지 - 개요

연구 방법

1. QoE 데이터 연구 및 AWS S3 기본 실습
2. AWS S3 에서 Dtv QoE 데이터 수집
3. Feature Selection 및 Feature Engineering
4. 실험
   - Resampling
   - SMOTE

이슈

1. QoE 데이터에 대한 이해 부족
2. 로컬 서버의 자원 이슈
3. 촉박한 기한

QoE 데이터를 이용한 이상징후 감지 – 이슈

1. QoE 데이터에 대한 이해 부족

– 생소한 QoE 데이터 지식 부재로 인해 전처리 방향 설정 불가

- **해결 방안**
  - 상급 담당자에게 QoE 관련 메타데이터와 QoE 데이터 생명 주기, 구조 등 안내 요청
  - 숙지한 내용을 바탕으로 팀원과 탐구
  - 탐구한 정보를 바탕으로 데이터 의미가 중복되거나 무의미한 정보 선별 규칙 수립 후 전처리 진행

Field Name	Playback Updated	Playback Ended
sessionId	O	O
bufferedDuration	O	
bufferings	O	
configurations		
contentUrl	O	O
currentPosition	O	
device	O	O
device.*	O	O
duration		
status		
timestamp	O	O

	Playback Updated	Playback Ended
errorCode		O
errorString		O
estimatedBandwidth	O	
event	O	O
fragmentSum.*	O	
fragments	O	
frameDropped	O	
licenseInfo.*	O	O
maxDecodingTime	O	
tracks		
userEvents	O	

	Playback Updated	Playback Ended
networkChanged	O	
networkErrors	O	
networkInfo.*	O	O
position		O
qoeContentId	O	O
qualityChanged	O	
redirectUrl	O	O
avgDecodingTime	O	
startupTime		
userInfo		

Data Merging Rule

sessionId	event	time stamp	C	D	E	F	G
123	playbackEnded	15632	c	d	e	f	g

sessionId	event	timestamp	C	D	K	L	M
123	playbackUpdated	15000	c	d	k	f	g
123	playbackUpdated	15123	a	d	b	e	g
123	playbackUpdated	15345	d	a	m	d	h
198	playbackUpdated	15589	t	j	f	n	m

sessionId	timestamp	C	D	E	F	G	K	L	M
123	15632	c	d	e	f	g	m	d	h

QoE 데이터를 이용한 이상징후 감지 - 이슈

2. 로컬 서버의 자원 이슈

- 데이터 분석 프로젝트를 진행하기에 열악한 컴퓨팅 자원 환경
- 팀원과 데이터를 분할하여 진행 시도 했지만 여전히 자원이 한정적
- 상황 보고 후 AWS EC2 인스턴스 요청 → 할당 받은 EC2도 자원의 한계로 시계열 데이터 예측 불가

- **해결 방안**
  - 실험 할 수 있는 데이터 용량의 최대치 산정 → 2일 치의 데이터로 분석 진행
  - 짧은 기간 데이터로 추세 파악이 불가하여 시계열 예측 모델 개발에서 분류 모델 개발로 방향 재설정

3. 촉박한 기한

- 서버 모니터링 시스템 개발 병행으로 인력 부족 → 개발 기한 촉박
- 분업 환경에서 무질서하고 각기 다른 개발 성향으로 인한 의견 차이 지속적으로 발생 → 진행 부진
- 상대 작업물에 대한 이해 시간 필요

- **해결 방안**
  - 매일 ToDo 리스트 공유하고 퇴근 전 진행상황을 서로 보고 후 반드시 결과물을 병합
  - 에러 발생이나 의문점 발생 시 서로 즉시 보고하여 크로스 체킹 하며 시간 단축

QoE 데이터를 이용한 이상징후 감지 - 결론

결론

- 자원 이슈와 데이터 탐색을 바탕으로 한 시계열 예측 불가 결론 도출 후 분류 모델 개발로 방향 재설정
- 로컬 서버에서 분류 모델 작업 완료

느낀 점

- 의사결정자의 추상적인 목표를 실현하기 위해 실무자 입장에서 접근, 설계 방식을 고려해 볼 수 있는 경험
- QoE 데이터를 통해 하나의 서비스가 완성도 있게 고객에게 제공되기 위해 수많은 feature들을 고려 해야 한다는 통찰을 얻을 수 있었음

  → 공급자는 다각도로 정밀하고 집요하게 데이터를 살펴보고 분석해야 한다는 인사이트
- 제한된 환경을 최대한 활용하여 설정한 목표에 근접하기 위해 때로는 우회도하며 새로운 인사이트를 얻기 위해 포기하지 않고 최선을 다하는 자세를 배움

아쉬웠던 점

- 좋은 데이터와 좋은 방향성을 가진 프로젝트가 자원 한계로 인해 불성사 → 회사 선택 기준에 인프라나 업무 지원 항목 확인 필수
- 제한된 기한 내 완성하기 위해 feature selection을 매뉴얼하게 진행

실무 적용 방안

- 미디어 QoE 데이터에 대한 경험
- 제한된 환경에서 최대한 목표에 근접하기 위한 행동을 경험

감사합니다 ☺