2025-2 모바일프로젝트 기말고사(20241519 조예성)

온 디바이스 모델 특징 4개: 개인정보 보호, 오프라인 작동, 지연 없음, 경량화

MLKit 은 개발자와 (Tensorflow Lite)사이의 번역기

(Tensorflow Lite)은 ML Kit이 실제 연산을 수행할 때 사용하는 엔진

MLKit 목표 4개: 개발자 접근성 향상, 모바일 최적화, 보안성 강화, 확장성 확보

MLKit은 온 디바이스와 클라우드 기반 기능을 모두 지원한다: **O**, X

텐서플로우를 모바일 환경에 맞게 경량화 한 실행 엔진: Tensorflow Lite
- 담당 역할: 추론

Tensorflow Lite 모델 실행 과정: 모델 변환, 디바이스 탑재, 추론 실행
구성 요소 3개: Tensorflow Model, Converter, Interpreter

Tensorflow Lite 모델 변환 과정: 학습, 변환, 최적화, 배포

Tensorflow Lite 핵심 특징: 경량화, 멀티 백엔드, 다양한 언어 및 플랫폼, 커스텀 모델 지원

ML Kit은 Tensorflow Lite 위에 구축된 (고수준 API)레이어

ML Kit 내부 과정: 입력 처리->API 호출 ->TFLite모델 로드 -> 모델 추론 -> 후처리 -> 결과 반환

ML Kit API 4영역: Vision, Natural Language, Generative, Custom Model

얼굴 탐지 API: Face Detection

```
// 얼굴 탐지를 위한 옵션 설정, FaceDetectorOptions.Builder() 객체로 설정
FaceDetectorOptions options = new FaceDetectorOptions.Builder().setPerformanceMode(FaceDetectorOptions.PERFORMANCE_MODE_FAST).build();

// 얼굴 탐지기 객체 생성, 설정한 옵션을 파리미터로 입력
FaceDetector detector = FaceDetection.getClient(options);
// InputImage 객체에 assets 폴더에서 가져온 이미지 연결
InputImage image = InputImage.fromBitmap(bitmap, 0);
// 탐지기 실행, 실행 결과 성공, 실패 리스너 설정 추가
detector.process(image).addOnSuccessListener(faces -> {
    // drawWithRectangle 메소드를 이용하여 성공시 이미지에 얼굴부분에 바운딩 박스 추가
    Bitmap out = drawWithRectangle(bitmap, faces);
    if(out != null){
        img.setImageBitmap(out);
    }
```

이미지 내 객체 식별 및 레이블링: Image Labeling
- implementation("com.google.mlkit:image-labeling:17.0.9")
- 신뢰도 임계 값 설정: Builder().setConfidenceThreshold(float).build()

이미지 속 여러 객체 위치 클래스(바운딩박스) 찾기: Object Detection
- implementation("com.google.mlkit:object-detection:17.0.2")
- 바운딩 박스 내 물체 판별: Image Labeling
- 옵션: setDectionMode(), enableMultipleObjects(), enableClassification(), setMaxPerObjectLabelCound()

```
InputImage image = InputImage.fromBitmap(bitmap, 0);
// 이미지 분류 및 레이블 찾기 API 권한 가져오기
ImageLabeler labeler = ImageLabeling.getClient(ImageLabelerOptions.DEFAULT_OPTIONS);
// 입력 이미지 추론 실행 및 결과 콜백 처리
labeler.process(image).addOnSuccessListener(new OnSuccessListener<List<ImageLabel>>() {
```

문장 내 특정 항목 추출: EntityExtractor

- 구성 요소 4개: Tokenizer, NER TFLite모델, Rule-based Normalizer, Structured Entity Builder

NER 구조: 문장 입력, 토큰화, 시퀀스 모델 입력, 각 단어 태그 예측,엔티티조합, 구조화된 엔티티 반환
- 문장을 단어 단위로 나누어 각 단에 택를 부탁하는 것: BIO/IOB 태깅

Implementation: Implementation("com.google.mlkit:entity-extraction:16.0.0-beta6")

```
entityExtractor.annotate(params).addOnSuccessListener(new OnSuccessListener<List<EntityAnnotation>>() {
            @Override
            public void onSuccess(List<EntityAnnotation> result) {
                // result: 추출된 엔터티의 목록
                StringBuilder sb = new StringBuilder();
                System.out.println(result);
                // EntityAnnotation 하나가
                // - 원문에서의 특정 구간(예: "내일 오후 3시")
                // - 그 구간에 해당하는 엔터티 리스트 (날짜/시간, 금액 등)를 포함하고 있음
                for (EntityAnnotation annotation : result) {
                    // 원문에서 엔터티가 발견된 부분 (예: "내일 오후 3시")
                    sb.append(annotation.getAnnotatedText());
                    // 해당 구간에 어떤 엔터티 타입들이 있는지 확인
                    for (Entity entity : annotation.getEntities()) {
                        sb.append(":").append(getStringFor(entity)); // 엔터티 타입을 문자열로 변환
                    }
                    sb.append("\n\n"); // 보기 좋게 줄바꿈
                }

                // 최종적으로 만들어진 문자열을 화면에 출력
                txtOutput.setText(sb.toString());
            }
```

사용자가 그린 획 데이터 분석하여 텍스트 또는 구조화된 결과로 변환하는: Digital Ink Recognition
- 동작: 문자 -> 단어 -> 문장

항목	Digital Ink	OCR(Text Recognition)
입력	좌표 스트로크 벡터	픽셀 이미지
영향 요소	배경 영향 없음	배경·노이즈 영향 큼
처리 방식	LSTM/Transformer	CNN 기반
장점	실시간·정확	사진 인식 가능
사용 사례	메모/필기	문서 촬영

ML Kit은 사용자의 손글씨를 (벡터 형태) 로 처리
- 포함 정보: x, y, t, stroke group

모델 내부 동작 방식 순서: 스토로크 정규화, 특성 벡터화, 디코딩 및 분류

디지털 잉크 장점: 벡터기반, 온디바이스, 다양한 언어 및 필기체, 낮은 지연시간

디지털 잉크 인식 모델 종류: 텍스트 인식, 도형 인식, 수학 모델, 스케치/드로잉 모델, 이모티콘모델

프롤세스 단계:
입력 이벤트 수집, 스트로크 그룹 생성, 스트로크 정규화, 특징 벡터화, 인식 모델 처리, 언어 모델 결합, 최종 결과 출력
- 이벤트 수집 구조: (x, y, timestamp, stroke_id)
- Pen down -> Pen Up을 하나의 (스트로크)로 정의
- 스트로크 정규화 4개: 크기 정규화, 시간 정규화, 노이즈 필터링, 속도 방향 추가계산

디지털 잉크 인식: implementation("com.google.mlkit:digital-ink-recognition:19.0.0")

```
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>
<uses-permission android:name="android.permission.INTERNET"/>
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
```

```
DigitalInkRecognitionModelIdentifier modelIdentifier;
try {
    // fromLanguageTag: 언어 코드로 Digital Ink 모델 식별자 생성
    modelIdentifier = DigitalInkRecognitionModelIdentifier.fromLanguageTag("en-US");
```

model = DigitalInkRecognitionModel.builder(modelIdentifier).build();

자동 답장 생성 API: SmartReply
- 지원 언어: duddj

동작 매커니즘: 대화 _> 전처리, 언어 판별 -> 안전 필터 -> 토큰화 -> Transformer -> 응답 벡터 유사도 계산_> 상위 3개 답장 선택

영어 아닐 경우: STATUS_NOT_SUPPORTED_LANGUAGE

민감한 주제일 경우: STATUS_NO_REPLY

implementation("com.google.mlkit:smart-reply:17.0.4")

```
// SmartReply 클라이언트 가져오기 (on-device 모델을 내부적으로 사용)
SmartReply.getClient() SmartReplyGenerator
        // 지금까지의 대화(conversation)를 입력으로 넣어 답장 후보 요청
        .suggestReplies(conversation) Task<SmartReplySuggestionResult>
        // 성공적으로 결과를 받았을 때
        .addOnSuccessListener(new OnSuccessListener<SmartReplySuggestionResult>() {
            no usages
            @Override
            public void onSuccess(SmartReplySuggestionResult result) {

                // result.getStatus()로 Smart Reply 동작 상태 확인
                if (result.getStatus() == SmartReplySuggestionResult.STATUS_SUCCESS) {
                    // STATUS_SUCCESS: 답장 후보를 정상적으로 생성한 경우

                    // 최대 3개의 SmartReplySuggestion 리스트가 반환됨
                    List<SmartReplySuggestion> suggestions = result.getSuggestions();
                    if (suggestions != null && !suggestions.isEmpty()) {
                        // 여기서는 가장 첫 번째 후보만 EditText에 자동으로 입력해 줌
                        txtInput.setText(suggestions.get(0).getText());
                    }
```

```
// → 최근 대화 기록들을 시간 순서대로 저장
private final ArrayList<TextMessage> conversation = new ArrayList<>();
```

```
SmartReply.getClient()
        // 지금까지의 대화(conversation)를 입력으로 넣어 답장 후보 요청
        .suggestReplies(conversation)
        // 성공적으로 결과를 받았을 때
        .addOnSuccessListener(new OnSuccessListener<SmartReplySuggestionResult>() {
```

대화 요소 추가: addConversationItem()

기능	메소드 / 옵션	설명
입력 이미지 생성	`InputImage.fromBitmap()`	ML Kit 표준 포맷으로 변환
라벨러 생성	`ImageLabeling.getClient(options)`	옵션 기반 라벨러 가져오기
기본 옵션	`ImageLabelerOptions.DEFAULT_OPTIONS`	사전 학습 On-device 모델 사용
결과 처리	`labeler.process(image)`	비동기 처리 → 성공/실패 콜백

기능	메소드 / 옵션	설명
옵션 설정	`new ObjectDetectorOptions.Builder()`	감지 모드 & 여러 객체 설정
객체 감지	`ObjectDetection.getClient(options)`	ObjectDetector 생성
이미지 처리	`detector.process(image)`	감지 결과 반환

기능	메소드	설명
Extractor 생성	`EntityExtraction.getClient(options)`	언어 옵션 필요
엔터티 추출	`extractor.annotate(text)`	비동기 추출 리스트 반환

기능	메소드	설명
모델 ID	`DigitalInkRecognitionModelIdentifier.fromLanguageTag("ko")`	한글, 영어, 일본어 등
모델 로드	`DigitalInkRecognitionModel.newBuilder(modelId)`	모델 생성
Recognizer 생성	`DigitalInkRecognizer.getClient(model)`	
스트로크 데이터 구성	`Ink.Builder()` / `Stroke.Builder()`	x,y,timestamp 추가
인식 실행	`recognizer.recognize(ink)`	후보 텍스트 반환

기능	메소드
SmartReply 생성	`SmartReply.getClient()`
답장 생성	`generator.suggestReplies(conversation)`

필드	설명
`getStatus()`	정상 / 언어 미지원 / 위험 콘텐츠
`getSuggestions()`	자동 생성 답장 후보 List
`SmartReplySuggestion.getText()`	"Sounds good!", "Thanks!" 등

분야	API	핵심 메소드	주요 반환값
이미지 분류	ImageLabeling	process()	List<ImageLabel>
객체 탐지	ObjectDetection	process()	List<DetectedObject>
텍스트 엔터티 추출	EntityExtractor	annotate()	List<EntityAnnotation>
손글씨 인식	Digital Ink	recognize()	RecognitionResult
자동 답장 생성	Smart Reply	suggestReplies()	SmartReplySuggestionResult