Navigation Log 분석 가이드

`navigation_logs` 테이블 데이터 분석을 위한 가이드 문서입니다.

1. 시간 칼럼 정의

예측 시간

칼럼	설명	계산 방식
`estimated_time_seconds`	예측 총 시간	**transit**: 보정된_보행시간 + 대중교통_탑승시간 **walking**: 보정된_보행시간 (전체가 도보)
`estimated_walk_time_seconds`	예측 보행 시간	`total_time_with_crosswalk` (factor 적용 + 횡단보도 1/3 대기)

실측 시간

칼럼	설명	계산 방식
`actual_time_seconds`	실제 총 시간	`ended_at - started_at`
`active_walking_time_seconds`	순수 걷기 시간	5초 이상 정지 구간 제외한 걷기만
`paused_time_seconds`	정지/대기 시간	신호대기, 환승, **대중교통 탑승 등 모두 포함**

시간 차이

칼럼	설명	계산 방식
`time_difference_seconds`	전체 시간 차이	`actual_time_seconds - estimated_time_seconds` (+: 늦음, -: 빠름)
`walk_time_difference_seconds`	도보 시간 차이	`active_walking_time_seconds - estimated_walk_time_seconds`

2. 정확도 칼럼 정의

칼럼	비교 대상	공식	의미
`accuracy_percent`	estimated vs actual	`100 - |actual - estimated| / estimated × 100`	전체 시간 예측 정확도
`walk_accuracy_percent`	estimated_walk vs active_walking	`100 - |active_walking - estimated_walk| / estimated_walk × 100`	**도보 예측 정확도 ★ 핵심 지표**

3. Factor 칼럼 정의

공식

```
최종시간 = 기준시간 × user_speed_factor × slope_factor × weather_factor
```

Factor 해석

Factor	계산 방식	< 1.0	= 1.0	> 1.0
`user_speed_factor`	`1.111 / user_speed_mps` (= 4km/h ÷ 사용자속도)	빠른 사용자	4km/h 사용자	느린 사용자
`slope_factor`	Tobler's Hiking Function	내리막 (빠름)	평지	오르막 (느림)
`weather_factor`	`1 / weather_coeff`	좋은 날씨	보통	나쁜 날씨

기준 속도 (1.111 m/s)

```
4 km/h ÷ 3.6 = 1.111 m/s
```

- TMap API의 기준 보행 속도: **4 km/h**
- 백엔드에서 거리를 4km/h로 나누어 기준 시간 재계산

Factor 예시

사용자 속도	user_speed_factor	해석
6 km/h	0.667	기준보다 50% 빠름, 시간 33% 단축
4 km/h	1.0	기준과 동일
3 km/h	1.333	기준보다 25% 느림, 시간 33% 증가

4. 비교 분석 방법

전체 시간 분석

```
예측: estimated_time_seconds
실측: actual_time_seconds
정확도: accuracy_percent
```

도보 시간 분석 ★ 핵심

```
예측: estimated_walk_time_seconds
실측: active_walking_time_seconds (순수 걷기)
정확도: walk_accuracy_percent
```

⚠ 주의사항

- **transit 모드**에서 `paused_time_seconds`에는 **지하철 탑승 시간도 포함**됨
- `paused_time ≠ 신호대기만` (대중교통 이용 시)

---

5. 속도 검증

real_walking_speed_kmh 검증

```
real_walking_speed_kmh = walking_distance_m ÷ active_walking_time_seconds × 3.6
```

user_speed_factor 역산

```
예측에 사용된 속도(m/s) = 1.111 ÷ user_speed_factor
예측에 사용된 속도(km/h) = 4.0 ÷ user_speed_factor
```

---

6. 분석 예시 (log_id=56)

기본 정보

항목	값
user_id	25
route_mode	transit
총 거리	15,294m
도보 거리	1,866m

Factor 분석

Factor	값	해석
user_speed_factor	0.657	4.0 ÷ 0.657 = **6.1 km/h** (빠른 사용자)
slope_factor	0.964	완만한 내리막 (3.6% 빠름)
weather_factor	1.014	약간 나쁜 날씨 (1.4% 느림)

estimated_time_seconds (2,840초) 구성

구성 요소	값	설명
보정된 보행시간	1,213초	factor 적용 + 횡단보도 1/3 대기

구성 요소	값	설명
대중교통 탑승시간	1,627초	TMap 제공 (지하철 이동)
합계	**2,840초**	

actual_time_seconds (2,980초) 구성

구성 요소	값	설명
순수 걷기	910초	active_walking_time
정지/대기	2,069초	신호대기 + 지하철 탑승/환승
합계	**2,980초**	

시간 비교 분석

구분	예측	실측	차이
보행 시간	1,213초	910초	-303초 (빠름)
대중교통+대기	1,627초	2,069초	+442초 (늦음)
총합	**2,840초**	**2,980초**	**+140초**

정확도

지표	값	평가
accuracy_percent	95.07%	우수 ✅
walk_accuracy_percent	75.02%	개선 필요 ⚠

분석 결론

- 사용자가 예측(6.1km/h)보다 더 빠르게(7.38km/h) 걸어서 303초 절약
- 대기/환승 시간이 예상보다 길어서 442초 추가
- 최종적으로 140초 늦음

---

7. 관련 파일

파일	설명
`backend/app/models.py`	NavigationLogs 모델 정의
`backend/app/schemas.py`	API 스키마 정의
`backend/app/routers/navigation_logs.py`	API 엔드포인트
`backend/app/utils/Factors_Affecting_Walking_Speed.py`	Factor 계산 로직
`frontend/services/navigationLogService.ts`	프론트엔드 로그 저장 로직

8. 핵심 요약

1. **우리 시스템의 핵심 지표**: `walk_accuracy_percent` (도보 예측 정확도)
2. **Factor < 1.0**: 시간이 단축됨 (빠름)
3. **Factor > 1.0**: 시간이 증가함 (느림)
4. **transit 모드 주의**: `paused_time`에 대중교통 탑승 시간 포함
5. **기준 속도**: 4 km/h (= 1.111 m/s)