ChatGPT

Improvement Plan for NAND Sequence Simulation and Generation

본 계획서는 `seq_sim.py` 기반 프로젝트의 목적을 유지하면서 시뮬레이터와 딥러닝 모델을 개선하기 위한 구체적인 설계 변경 사항을 정리한 문서입니다. 아래 내용은 여러 버전으로 코드를 분리하여 실험할 수 있도록 제안합니다.

1. 시뮬레이터 개선

1. **시나리오 정의** – 랜덤 시퀀스만 생성하던 기존 설계를 확장하여 다음과 같은 대표 시나리오를 명시적으로 생성합니다.
2. 연속 프로그램 패턴 : 하나의 블록에 대해 `erase` 후 여러 번 `program` 명령을 순차적으로 수행하고, 일정 간격마다 `read`를 삽입합니다.
3. GC(garbage collection) 루프 : 블록이 가득 차면 새로운 블록으로 전환하고, 이전 블록을 `erase`하는 과정을 반복합니다.
4. 랜덤 읽기 패턴 : `program`된 페이지들 중 무작위로 `read`를 수행하되, `read_offset_limit`을 준수하도록 합니다.

5. 각 시나리오를 구분할 수 있는 레이블을 함께 저장하여 데이터 분석 및 조건부 학습에 활용합니다.

6. **추가 특징** – 페이지 이동의 상대적 변화가 모델 학습에 중요합니다. 기존 절대 페이지 값 외에 다음과 같은 추가 피처를 도입합니다.

7. `delta_program_page` : 프로그램 명령일 때 현재 페이지와 직전 페이지의 차이 (`current_page - last_page`).
8. `delta_read_page` : 읽기 명령일 때 최근 프로그래밍된 페이지와 현재 읽는 페이지의 차이 (`last_program_page - current_page`).

9. 이렇게 상대적 오프셋을 포함하면 모델이 페이지 순서 제약을 더 쉽게 학습할 수 있습니다.

10. **희귀 패턴 생성** – 실제 환경에서 드물게 발생하는 오류나 특정 패턴을 데이터에 충분히 포함시키기 위해 시뮬레이터에 옵션을 추가하여 희귀 시나리오를 의도적으로 생성할 수 있도록 합니다.

위 개선사항을 반영하여 기존 `ContrastiveDataGenerator`를 확장하거나 새로운 시뮬레이터 클래스를 작성합니다. 이 변경은 모든 버전의 모델에서 공통으로 활용됩니다.

2. 모델 아키텍처 개선

개선된 요구 사항에 따라 여러 아키텍처 버전을 구현하고 비교합니다. 각 버전은 별도의 파이썬 파일로 작성되어 독립적으로 실행할 수 있도록 합니다.

2.1 기본 GRU 버전 (Baseline)

- 기존 `seq_sim.py`의 GRU 기반 인코더–디코더 구조를 유지합니다. 개선된 시뮬레이터와 추가 피처를 사용하여 베이스라인 성능을 측정합니다.

2.2 Transformer 기반 버전 (`seq_sim_transformer.py`)

- **인코더** : `nn.TransformerEncoder`를 사용하여 전체 시퀀스를 처리합니다. 입력 피처를 선형 투영하여 지정된 `embedding_dim` 크기의 벡터로 변환한 뒤 positional encoding을 적용합니다.
- **다음 이벤트 예측** : 디코더를 별도로 두지 않고, 인코더의 마지막 타임스텝 표현을 사용해 다음 명령과 주소를 직접 예측합니다. 이는 다음 두 개의 헤드로 구성됩니다.
- 명령 헤드: `nn.Linear`를 거쳐 softmax 로짓을 출력하고 `CrossEntropyLoss`로 학습합니다.
- 주소 헤드: `nn.Linear`를 거쳐 연속적인 4차원 벡터를 출력하고 `MSELoss` 또는 Huber loss로 학습합니다.
- **훈련 전략** : 향후 실험을 위해 teacher forcing, 스케줄드 샘플링, professor forcing 등을 옵션으로 선택할 수 있게 설계합니다. 기본 구현에서는 teacher forcing을 사용합니다.

2.3 계층적/포인터 버전 (`seq_sim_pointer.py`)

- **계층적 분류** : 명령을 예측한 후 die, plane, block, page를 단계적으로 분류합니다. 페이지 값은 절대값 대신 `delta_program_page`나 `delta_read_page` 같은 상대적 오프셋 클래스로 예측할 수 있습니다.
- **포인터 네트워크** : 현재 블록의 프로그램된 페이지 집합을 기억하고, 어텐션을 통해 다음 페이지 위치를 선택하도록 설계합니다. 이는 주소 공간이 큰 경우에 유리합니다.
- 이 버전은 구현 복잡도가 높으므로 초기에는 계층적 분류와 포인터 네트워크를 위한 틀만 제공하고, 차후 실험에서 세부 구현을 추가합니다.

3. 훈련 및 평가 전략

1. **스케줄드 샘플링** – 학습 초기에는 teacher forcing 확률을 높게 유지하고, 에폭이 진행될수록 모델의 예측을 입력으로 사용하도록 확률을 감소시키는 스케줄을 구현합니다. 이는 노출 편향을 완화하는 데 도움이 됩니다.
2. **Professor Forcing (옵션)** – 추가 실험으로, 모델이 학습 단계와 샘플링 단계에서 동일한 동적 특성을 갖도록 적대적 도메인 적응을 적용하는 버전을 구현할 수 있습니다 [1].
3. **손실 함수** – 주소 예측에는 Huber loss를 사용하여 큰 오차의 영향을 완화하고, 명령 예측에는 class imbalance를 고려해 focal loss를 옵션으로 제공합니다 [2].
4. **평가 지표** – 기존 유효성 검사기(validator)를 활용해 생성된 시퀀스의 규칙 준수율을 측정하고, 시나리오별 정확도와 길이별 성공률을 분석합니다.

4. 파일 구조 제안

파일명	내용
`seq_sim_baseline.py`	개선된 시뮬레이터와 추가 피처(`delta_program_page`, `delta_read_page`)를 사용하여 GRU 기반으로 다음 명령과 주소를 예측하는 베이스라인 코드입니다. `ContrastiveDataGenerator`로 생성한 시퀀스에 대해 슬라이딩 윈도우 방식의 데이터셋을 만들고, `GRUModel`을 통해 명령과 주소를 동시에 회귀합니다.
`seq_sim_transformer.py`	Transformer 기반 next-step 예측 모델 구현. 입력 시퀀스를 임베딩하여 `nn.TransformerEncoder`로 처리하고, 마지막 타임스텝 표현을 이용해 다음 명령과 주소를 출력합니다. 스케줄드 샘플링, Huber loss 등을 옵션으로 지원하도록 설계할 수 있습니다.
`seq_sim_pointer.py`	계층적/포인터 네트워크 구조의 틀을 제시하는 코드 (추후 확장 가능). 초기 버전에서는 구조적 설계와 TODO를 포함한 스켈레톤을 제공하며, 향후 연구 단계에서 구체적으로 구현합니다.

파일명	내용
`improvement_plan.md`	본 계획서. 새로운 버전의 구현 세부사항과 향후 추가할 기능에 대한 설명을 포함합니다.

각 파일은 공통적인 데이터 생성 및 특징 추출 모듈을 import 하여 재사용성을 높입니다.

---

1 [1610.09038] Professor Forcing: A New Algorithm for Training Recurrent Networks
https://arxiv.org/abs/1610.09038

2 [1708.02002] Focal Loss for Dense Object Detection
https://arxiv.org/abs/1708.02002