**Binclone Python 구현체**

1. **설치**

기본적으로 파이선 3.7.8 이상의 버전이 필요하며, pandas 1.2.5 설치가 필요하다.

1. **사용법**

2 가지 모듈이 있는데, BinCloneNaive 와 BinCloneFunctionWise이다. Normalizer는 따로 사용할 대상은 아니다.

* 1. **BinCloneNaive**

이 모듈은 함수명에 대한 정보가 없는 상태의 모듈이다. 사용자는

*generate\_database*

*load\_database*

*search\_query*

의 함수들을 사용할 수 있다.

우선 클래스의 선언을 한 후, 각 함수를 호출하여 선언 가능하다. search\_query를 제외하곤 inplace 함수들이다 (None 을 return)

binclone = BinCloneNaive(region\_window=60, SBSize=24)

binclone.generate\_database(["asms/open18\_sort.S", "asms/open18\_disrupt.S"], "db.csv")

binclone.load\_database("db.csv")

df = binclone.search\_query("query.S")

print(df)

같은 예시 케이스를 두고 생각하겠다.

Initialization:

사용자는 데이터베이스화 할 때, region 을 얼마나 크게 둘 지, 사용할 feature 개수는 몇 개인지 염두해야 한다. 이는 initialize 할 때 지정 가능하다.

**generate\_database:**

: 데이터베이스를 입력받은 asm 파일들을 기반으로 생성함.

* 논문에서 나왔듯, 중앙값을 기준으로 0 이하인 feature 들은 버리고, 중앙값이 되는 feature 개수를 정해줘야 하므로 이를 input으로 받은 asm파일들을 계산한다. Med\_bounds.json 이란 파일을 경로에 저장한다.
* 후에 데이터베이스를 만들고 db.csv 와 같은 사용자 지정 파일 경로에 저장한다.

**load\_database:**

: 데이터베이스를 로드함.

* 입력 받은 위치의 csv를 읽어온다. Binclone 내부에 데이터베이스를 저장한다.

**search\_query:**

: 로드한 데이터베이스에서 쿼리 파일과 가장 비슷한 asm 의 region 들을 리턴함.

* 입력으로 받은 쿼리의 상위 region\_window 수만큼의 줄을 기반으로 쿼리 벡터를 생성하여, 이 벡터와 유사도가 높은 결과를 리턴한다.
* Pd.DataFrame 을 리턴하며, 이는 filepath, idx attribute 가 있는데,

Filepath 는 가장 유사한 region 이 위치한 asm 파일 경롤이고, idx 는 그 파일에서 몇번째 줄인지를 의미한다.

위 코드 실행의 예시 output :

텍스트이(가) 표시된 사진

자동 생성된 설명

* 1. **BinCloneFunctionWise**

이 모듈은 추가적으로 가장 유사한 region 이 어떤 함수에서 왔는지도 알려주는 모듈이다. 사용자는

*generate\_database*

*load\_database*

*search\_query*

을 사용할 수 있다.

binclone = BinCloneFunctionWise(region\_window=12, SBSize=5, percentile=0.8)

binclone.generate\_database(glob("./original/\*.txt"), db="db.csv")

binclone.load\_database("db.csv")

df = binclone.search\_query("./queries/t100.txt")

print(df)

같은 예시 케이스를 두고 생각하겠다.

**Initialization:**

사용자는 데이터베이스화 할 때, region 을 얼마나 크게 둘 지, 사용할 feature 개수는 몇 개인지 염두해야 한다. 이는 initialize 할 때 지정 가능하다. 추가적으로, 논문에선 없었지만 어떤 케이스는 window size 가 너무 작기에 median feature 가 0 인 것이 너무 많았다. 때문에, 50% percentile 이상의 값을 기준으로 feature boundary 를 정하고 싶을 수도 있다. 이에, percentile 을 지정하여 binclone 객체를 선언할 수 있다.

**generate\_database :**

: 데이터베이스를 입력받은 asm 파일들을 기반으로 생성함.

* 논문에서 나왔듯, 중앙값을 기준으로 0 이하인 feature 들은 버리고, 중앙값이 되는 feature 개수를 정해줘야 하므로 이를 input으로 받은 asm 파일들을 계산한다. med\_bounds.json 이란 파일을 경로에 저장한다.
* 후에 데이터베이스를 만들고 db.csv 와 같은 사용자 지정 파일 경로에 저장한다.

**load\_database:**

: 데이터베이스를 로드함.

* 입력 받은 위치의 csv를 읽어온다. Binclone 내부에 데이터베이스를 저장한다.

**search\_query:**

: 로드한 데이터베이스에서 쿼리 파일과 가장 비슷한 asm 의 region들을 리턴함.

* 입력으로 받은 쿼리의 상위 region\_window 수만큼의 줄을 기반으로 쿼리 벡터를 생성하여, 이 벡터와 유사도가 높은 결과를 리턴한다.
* Pd.DataFrame 을 리턴하며, 이는 filepath, functionname, idx attribute 가 있는데,

filepath는 가장 유사한 region 이 위치한 asm 파일 경로이고, function\_name 은 그 asm 파일에서 어떤 함수 (또, 그 함수가 몇 번째 줄에 위치해 있는지도), idx 는 그 함수내부에서 몇 번째 줄인지 알려준다.

위 코드의 예시 실행 결과 :

텍스트이(가) 표시된 사진

자동 생성된 설명