use anyhow::Result;

use log::info;

use navi::cli\_args::{ARGS, MODEL\_SPECS};

use navi::cores::validator::validatior::cli\_validator;

use navi::tf\_model::tf::TFModel;

use navi::{bootstrap, metrics};

use sha256::digest;

fn main() -> Result<()> {

env\_logger::init();

cli\_validator::validate\_input\_args();

//only validate in for tf as other models don't have this

assert\_eq!(MODEL\_SPECS.len(), ARGS.serving\_sig.len());

metrics::register\_custom\_metrics();

//load all the custom ops - comma seperaed

if let Some(ref customops\_lib) = ARGS.customops\_lib {

for op\_lib in customops\_lib.split(",") {

load\_custom\_op(op\_lib);

}

}

// versioning the customop so library

bootstrap::bootstrap(TFModel::new)

}

fn load\_custom\_op(lib\_path: &str) -> () {

let res = tensorflow::Library::load(lib\_path);

info!("{} load status:{:?}", lib\_path, res);

let customop\_version\_num = get\_custom\_op\_version(lib\_path);

// Last OP version is recorded

metrics::CUSTOMOP\_VERSION.set(customop\_version\_num);

}

//fn get\_custom\_op\_version(customops\_lib: &String) -> i64 {

fn get\_custom\_op\_version(customops\_lib: &str) -> i64 {

let customop\_bytes = std::fs::read(customops\_lib).unwrap(); // Vec<u8>

let customop\_hash = digest(customop\_bytes.as\_slice());

//conver the last 4 hex digits to version number as prometheus metrics doesn't support string, the total space is 16^4 == 65536

let customop\_version\_num =

i64::from\_str\_radix(&customop\_hash[customop\_hash.len() - 4..], 16).unwrap();

info!(

"customop hash: {}, version\_number: {}",

customop\_hash, customop\_version\_num

);

customop\_version\_num

}