use serde::{Deserialize, Serialize};

use serde\_json::Error;

#[derive(Default, Debug, Clone, PartialEq, Serialize, Deserialize)]

#[serde(rename\_all = "camelCase")]

pub struct AllConfig {

#[serde(rename = "train\_data")]

pub train\_data: TrainData,

}

#[derive(Default, Debug, Clone, PartialEq, Serialize, Deserialize)]

#[serde(rename\_all = "camelCase")]

pub struct TrainData {

#[serde(rename = "seg\_dense\_schema")]

pub seg\_dense\_schema: SegDenseSchema,

}

#[derive(Default, Debug, Clone, PartialEq, Serialize, Deserialize)]

#[serde(rename\_all = "camelCase")]

pub struct SegDenseSchema {

#[serde(rename = "renamed\_features")]

pub renamed\_features: RenamedFeatures,

}

#[derive(Default, Debug, Clone, PartialEq, Serialize, Deserialize)]

#[serde(rename\_all = "camelCase")]

pub struct RenamedFeatures {

pub continuous: String,

pub binary: String,

pub discrete: String,

#[serde(rename = "author\_embedding")]

pub author\_embedding: String,

#[serde(rename = "user\_embedding")]

pub user\_embedding: String,

#[serde(rename = "user\_eng\_embedding")]

pub user\_eng\_embedding: String,

#[serde(rename = "meta\_\_author\_id")]

pub meta\_author\_id: String,

#[serde(rename = "meta\_\_user\_id")]

pub meta\_user\_id: String,

#[serde(rename = "meta\_\_tweet\_id")]

pub meta\_tweet\_id: String,

}

pub fn parse(json\_str: &str) -> Result<AllConfig, Error> {

let all\_config: AllConfig = serde\_json::from\_str(json\_str)?;

Ok(all\_config)

}