package com.twitter.tweetypie

package repository

import com.ibm.icu.util.ULocale

import com.twitter.common.text.pipeline.TwitterLanguageIdentifier

import com.twitter.stitch.Stitch

import com.twitter.stitch.compat.LegacySeqGroup

import com.twitter.tweetypie.repository.LanguageRepository.Text

import com.twitter.tweetypie.thriftscala.\_

import com.twitter.util.FuturePool

import com.twitter.util.logging.Logger

object LanguageRepository {

type Type = Text => Stitch[Option[Language]]

type Text = String

}

object PenguinLanguageRepository {

private val identifier = new TwitterLanguageIdentifier.Builder().buildForTweet()

private val log = Logger(getClass)

def isRightToLeft(lang: String): Boolean =

new ULocale(lang).getCharacterOrientation == "right-to-left"

def apply(futurePool: FuturePool): LanguageRepository.Type = {

val identifyOne =

FutureArrow[Text, Option[Language]] { text =>

futurePool {

try {

Some(identifier.identify(text))

} catch {

case e: IllegalArgumentException =>

val userId = TwitterContext().map(\_.userId)

val encodedText = com.twitter.util.Base64StringEncoder.encode(text.getBytes)

log.info(s"${e.getMessage} : USER ID - $userId : TEXT - $encodedText")

None

}

}.map {

case Some(langWithScore) =>

val lang = langWithScore.getLocale.getLanguage

Some(

Language(

language = lang,

rightToLeft = isRightToLeft(lang),

confidence = langWithScore.getScore

))

case None => None

}

}

text => Stitch.call(text, LegacySeqGroup(identifyOne.liftSeq))

}

}