package com.twitter.simclusters\_v2.summingbird.common

import com.twitter.recos.entities.thriftscala.NamedEntity

import com.twitter.simclusters\_v2.thriftscala.{

NerKey,

PenguinKey,

SimClusterEntity,

TweetTextEntity

}

import com.twitter.taxi.util.text.{TweetFeatureExtractor, TweetTextFeatures}

import com.twitter.tweetypie.thriftscala.Tweet

object TweetEntityExtractor {

private val MaxHashtagsPerTweet: Int = 4

private val MaxNersPerTweet: Int = 4

private val MaxPenguinsPerTweet: Int = 4

private val tweetFeatureExtractor: TweetFeatureExtractor = TweetFeatureExtractor.Default

private def extractTweetTextFeatures(

text: String,

languageCode: Option[String]

): TweetTextFeatures = {

if (languageCode.isDefined) {

tweetFeatureExtractor.extract(text, languageCode.get)

} else {

tweetFeatureExtractor.extract(text)

}

}

def extractEntitiesFromText(

tweet: Option[Tweet],

nerEntitiesOpt: Option[Seq[NamedEntity]]

): Seq[SimClusterEntity.TweetEntity] = {

val hashtagEntities = tweet

.flatMap(\_.hashtags.map(\_.map(\_.text))).getOrElse(Nil)

.map { hashtag => TweetTextEntity.Hashtag(hashtag.toLowerCase) }.take(MaxHashtagsPerTweet)

val nerEntities = nerEntitiesOpt

.getOrElse(Nil).map { namedEntity =>

TweetTextEntity

.Ner(NerKey(namedEntity.namedEntity.toLowerCase, namedEntity.entityType.getValue))

}.take(MaxNersPerTweet)

val nerEntitySet = nerEntities.map(\_.ner.textEntity).toSet

val penguinEntities =

extractTweetTextFeatures(

tweet.flatMap(\_.coreData.map(\_.text)).getOrElse(""),

tweet.flatMap(\_.language.map(\_.language))

).phrases

.map(\_.normalizedOrOriginal)

.filter { s =>

s.charAt(0) != '#' && !nerEntitySet.contains(s) // not included in hashtags and NER

}

.map { penguinStr => TweetTextEntity.Penguin(PenguinKey(penguinStr.toLowerCase)) }.take(

MaxPenguinsPerTweet)

(hashtagEntities ++ penguinEntities ++ nerEntities).map(e => SimClusterEntity.TweetEntity(e))

}

}