package com.twitter.tweetypie.thriftscala.entities

import com.twitter.servo.data.Mutation

import com.twitter.tco\_util.TcoUrl

import com.twitter.tweetypie.thriftscala.\_

import com.twitter.tweetypie.thriftscala.entities.Implicits.\_

import com.twitter.tweetypie.tweettext.PartialHtmlEncoding

import com.twitter.tweetypie.tweettext.TextEntity

import com.twitter.tweetypie.tweettext.TextModification

import com.twitter.tweetypie.util.TweetLenses

import com.twitter.twittertext.Extractor

import scala.collection.JavaConverters.\_

/\*\*

\* Contains functions to collect urls, mentions, hashtags, and cashtags from the text of tweets and messages

\*/

object EntityExtractor {

// We only use one configuration of com.twitter.twittertext.Extractor, so it's

// OK to share one global reference. The only available

// configuration option is whether to extract URLs without protocols

// (defaults to true)

private[this] val extractor = new Extractor

// The twitter-text library operates on unencoded text, but we store

// and process HTML-encoded text. The TextModification returned

// from this function contains the decoded text which we will operate on,

// but also provides us with the ability to map the indices on

// the twitter-text entities back to the entities on the encoded text.

private val htmlEncodedTextToEncodeModification: String => TextModification =

text =>

PartialHtmlEncoding

.decodeWithModification(text)

.getOrElse(TextModification.identity(text))

.inverse

private[this] val extractAllUrlsFromTextMod: TextModification => Seq[UrlEntity] =

extractUrls(false)

val extractAllUrls: String => Seq[UrlEntity] =

htmlEncodedTextToEncodeModification.andThen(extractAllUrlsFromTextMod)

private[this] val extractTcoUrls: TextModification => Seq[UrlEntity] =

extractUrls(true)

private[this] def extractUrls(tcoOnly: Boolean): TextModification => Seq[UrlEntity] =

mkEntityExtractor[UrlEntity](

extractor.extractURLsWithIndices(\_).asScala.filter { e =>

if (tcoOnly) TcoUrl.isTcoUrl(e.getValue) else true

},

UrlEntity(\_, \_, \_)

)

private[this] val extractMentionsFromTextMod: TextModification => Seq[MentionEntity] =

mkEntityExtractor[MentionEntity](

extractor.extractMentionedScreennamesWithIndices(\_).asScala,

MentionEntity(\_, \_, \_)

)

val extractMentions: String => Seq[MentionEntity] =

htmlEncodedTextToEncodeModification.andThen(extractMentionsFromTextMod)

private[this] val extractHashtagsFromTextMod: TextModification => Seq[HashtagEntity] =

mkEntityExtractor[HashtagEntity](

extractor.extractHashtagsWithIndices(\_).asScala,

HashtagEntity(\_, \_, \_)

)

val extractHashtags: String => Seq[HashtagEntity] =

htmlEncodedTextToEncodeModification.andThen(extractHashtagsFromTextMod)

private[this] val extractCashtagsFromTextMod: TextModification => Seq[CashtagEntity] =

mkEntityExtractor[CashtagEntity](

extractor.extractCashtagsWithIndices(\_).asScala,

CashtagEntity(\_, \_, \_)

)

val extractCashtags: String => Seq[CashtagEntity] =

htmlEncodedTextToEncodeModification.andThen(extractCashtagsFromTextMod)

private[this] def mkEntityExtractor[E: TextEntity](

extract: String => Seq[Extractor.Entity],

construct: (Short, Short, String) => E

): TextModification => Seq[E] =

htmlEncodedMod => {

val convert: Extractor.Entity => Option[E] =

e =>

for {

start <- asShort(e.getStart.intValue)

end <- asShort(e.getEnd.intValue)

if e.getValue != null

res <- htmlEncodedMod.reindexEntity(construct(start, end, e.getValue))

} yield res

val entities = extract(htmlEncodedMod.original)

extractor.modifyIndicesFromUTF16ToUnicode(htmlEncodedMod.original, entities.asJava)

entities.map(convert).flatten

}

private[this] def asShort(i: Int): Option[Short] =

if (i.isValidShort) Some(i.toShort) else None

private[this] def mutation(extractUrls: Boolean): Mutation[Tweet] =

Mutation { tweet =>

val htmlEncodedMod = htmlEncodedTextToEncodeModification(TweetLenses.text.get(tweet))

Some(

tweet.copy(

urls = if (extractUrls) Some(extractTcoUrls(htmlEncodedMod)) else tweet.urls,

mentions = Some(extractMentionsFromTextMod(htmlEncodedMod)),

hashtags = Some(extractHashtagsFromTextMod(htmlEncodedMod)),

cashtags = Some(extractCashtagsFromTextMod(htmlEncodedMod))

)

)

}

val mutationWithoutUrls: Mutation[Tweet] = mutation(false)

val mutationAll: Mutation[Tweet] = mutation(true)

}