package com.twitter.ann.common

import com.twitter.util.Return

import com.twitter.util.Throw

import com.twitter.util.Try

import com.twitter.util.logging.Logging

// Memoization with a twist

// New epoch reuse K:V pairs from previous and recycle everything else

class MemoizedInEpochs[K, V](f: K => Try[V]) extends Logging {

private var memoizedCalls: Map[K, V] = Map.empty

def epoch(keys: Seq[K]): Seq[V] = {

val newSet = keys.toSet

val keysToBeComputed = newSet.diff(memoizedCalls.keySet)

val computedKeysAndValues = keysToBeComputed.map { key =>

info(s"Memoize ${key}")

(key, f(key))

}

val keysAndValuesAfterFilteringFailures = computedKeysAndValues

.flatMap {

case (key, Return(value)) => Some((key, value))

case (key, Throw(e)) =>

warn(s"Calling f for ${key} has failed", e)

None

}

val keysReusedFromLastEpoch = memoizedCalls.filterKeys(newSet.contains)

memoizedCalls = keysReusedFromLastEpoch ++ keysAndValuesAfterFilteringFailures

debug(s"Final memoization is ${memoizedCalls.keys.mkString(", ")}")

keys.flatMap(memoizedCalls.get)

}

def currentEpochKeys: Set[K] = memoizedCalls.keySet

}