package com.twitter.search.common.query;

import java.util.IdentityHashMap;

import com.google.common.base.Preconditions;

import com.google.common.collect.Maps;

import com.twitter.search.queryparser.query.BooleanQuery;

import com.twitter.search.queryparser.query.Query;

import com.twitter.search.queryparser.query.QueryParserException;

import com.twitter.search.queryparser.query.annotation.Annotation;

import com.twitter.search.queryparser.visitors.DetectAnnotationVisitor;

/\*\*

\* A visitor that collects node ranks from :r annotation in the query

\*/

public class QueryRankVisitor extends DetectAnnotationVisitor {

private final IdentityHashMap<Query, Integer> nodeToRankMap = Maps.newIdentityHashMap();

public QueryRankVisitor() {

super(Annotation.Type.NODE\_RANK);

}

@Override

protected boolean visitBooleanQuery(BooleanQuery query) throws QueryParserException {

if (query.hasAnnotationType(Annotation.Type.NODE\_RANK)) {

collectNodeRank(query.getAnnotationOf(Annotation.Type.NODE\_RANK).get(), query);

}

boolean found = false;

for (Query child : query.getChildren()) {

found |= child.accept(this);

}

return found;

}

@Override

protected boolean visitQuery(Query query) throws QueryParserException {

if (query.hasAnnotationType(Annotation.Type.NODE\_RANK)) {

collectNodeRank(query.getAnnotationOf(Annotation.Type.NODE\_RANK).get(), query);

return true;

}

return false;

}

private void collectNodeRank(Annotation anno, Query query) {

Preconditions.checkArgument(anno.getType() == Annotation.Type.NODE\_RANK);

int rank = (Integer) anno.getValue();

nodeToRankMap.put(query, rank);

}

public IdentityHashMap<Query, Integer> getNodeToRankMap() {

return nodeToRankMap;

}

}