LIRe 源代码分析 3：基本接口（ImageSearcher）

2013年10月31日 20:48:59　阅读数：5673

==

LIRe源代码分析系列文章列表：

==

上篇文章介绍了LIRe源代码里的DocumentBuilder的几个基本接口。本文继续研究一下源代码里的ImageSearcher的几个基本接口。

下面来看看与ImageSearcher相关的类的定义：

ImageSearcher：接口，定义了基本的方法。

AbstractImageSearcher：纯虚类，实现了ImageSearcher接口。

ImageSearcherFactory：用于创建ImageSearcher。

ImageSearcher相关的类的继承关系如下图所示。可见，各种算法类都继承了AbstractImageSearcher，而AbstractImageSearcher实现了ImageSearcher接口。

此外还有一个结构体：

ImageSearchHits：用于存储搜索的结果。

详细的源代码如下所示：

ImageSearcher

[java]

```java
/*
 * This file is part of the LIRe project: http://www.semanticmetadata.net/lire
 * LIRe is free software; you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation; either version 2 of the License, or
 * (at your option) any later version.
 *
 * LIRe is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with LIRe; if not, write to the Free Software
 * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA
 *
 * We kindly ask you to refer the following paper in any publication mentioning Lire:
 *
 * Lux Mathias, Savvas A. Chatzichristofis. Lire: Lucene Image Retrieval 鈥�
 * An Extensible Java CBIR Library. In proceedings of the 16th ACM International
 * Conference on Multimedia, pp. 1085-1088, Vancouver, Canada, 2008
 *
 * http://doi.acm.org/10.1145/1459359.1459577
 *
 * Copyright statement:
 * --------------------
 * (c) 2002-2011 by Mathias Lux (mathias@juggle.at)
 *     http://www.semanticmetadata.net/lire
 */

package net.semanticmetadata.lire;

import org.apache.lucene.document.Document;
import org.apache.lucene.index.IndexReader;

import java.awt.image.BufferedImage;
import java.io.IOException;
import java.io.InputStream;
import java.util.Set;

/**
 * <h2>Searching in an Index</h2>
 * Use the ImageSearcherFactory for creating an ImageSearcher, which will retrieve the images
 * for you from the index.
 * <p/>
 * <pre>
 * IndexReader reader = IndexReader.open(indexPath);
 * ImageSearcher searcher = ImageSearcherFactory.createDefaultSearcher();
 * FileInputStream imageStream = new FileInputStream("image.jpg");
 * BufferedImage bimg = ImageIO.read(imageStream);
 * // searching for an image:
 * ImageSearchHits hits = null;
 * hits = searcher.search(bimg, reader);
 * for (int i = 0; i < 5; i++) {
 * System.out.println(hits.score(i) + ": " + hits.doc(i).getField(DocumentBuilder.FIELD_NAME_IDENTIFIER).stringValue());
 * }
 *
 * // searching for a document:
 * Document document = hits.doc(0);
 * hits = searcher.search(document, reader);
 * for (int i = 0; i < 5; i++) {
 * System.out.println(hits.score(i) + ": " + hits.doc(i).getField(DocumentBuilder.FIELD_NAME_IDENTIFIER).stringValue());
 * }
 * </pre>
 * <p/>
 * This file is part of the Caliph and Emir project: http://www.SemanticMetadata.net
 * <br>Date: 01.02.2006
 * <br>Time: 00:09:42
 *
 * @author Mathias Lux, mathias@juggle.at
 */
public interface ImageSearcher {
    /**
     * Searches for images similar to the given image.
     *
     * @param image  the example image to search for.
     * @param reader the IndexReader which is used to dsearch through the images.
     * @return a sorted list of hits.
     * @throws java.io.IOException in case exceptions in the reader occurs
     */
    public ImageSearchHits search(BufferedImage image, IndexReader reader) throws IOException;

    /**
     * Searches for images similar to the given image, defined by the Document from the index.
     *
     * @param doc    the example image to search for.
     * @param reader the IndexReader which is used to dsearch through the images.
     * @return a sorted list of hits.
     * @throws java.io.IOException in case exceptions in the reader occurs
     */
    public ImageSearchHits search(Document doc, IndexReader reader) throws IOException;

    /**
     * Searches for images similar to the given image.
     *
     * @param image  the example image to search for.
     * @param reader the IndexReader which is used to dsearch through the images.
     * @return a sorted list of hits.
     * @throws IOException in case the image could not be read from stream.
     */
    public ImageSearchHits search(InputStream image, IndexReader reader) throws IOException;

    /**
     * Identifies duplicates in the database.
     *
     * @param reader the IndexReader which is used to dsearch through the images.
     * @return a sorted list of hits.
     * @throws IOException in case the image could not be read from stream.
     */
    public ImageDuplicates findDuplicates(IndexReader reader) throws IOException;

    /**
     * Modifies the given search by the provided positive and negative examples. This process follows the idea
     * of relevance feedback.
     *
     * @param originalSearch
     * @param positives
     * @param negatives
     * @return
     * @return
     */
    public ImageSearchHits relevanceFeedback(ImageSearchHits originalSearch,
                                             Set<Document> positives, Set<Document> negatives);
}
```

从接口的源代码可以看出，提供了5个方法，其中有3个名字都叫search()，只是参数不一样。一个是BufferedImage，一个是Document，而另一个是InputStream。

AbstractImageSearcher

[java]

```java
/*
 * This file is part of the LIRe project: http://www.semanticmetadata.net/lire
 * LIRe is free software; you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation; either version 2 of the License, or
 * (at your option) any later version.
 *
 * LIRe is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with LIRe; if not, write to the Free Software
 * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA
 *
 * We kindly ask you to refer the following paper in any publication mentioning Lire:
 *
 * Lux Mathias, Savvas A. Chatzichristofis. Lire: Lucene Image Retrieval 鈥�
 * An Extensible Java CBIR Library. In proceedings of the 16th ACM International
 * Conference on Multimedia, pp. 1085-1088, Vancouver, Canada, 2008
 *
 * http://doi.acm.org/10.1145/1459359.1459577
 *
 * Copyright statement:
 * --------------------
 * (c) 2002-2011 by Mathias Lux (mathias@juggle.at)
 *     http://www.semanticmetadata.net/lire
 */
package net.semanticmetadata.lire;

import org.apache.lucene.document.Document;
import org.apache.lucene.index.IndexReader;

import javax.imageio.ImageIO;
import java.awt.image.BufferedImage;
import java.io.IOException;
import java.io.InputStream;
import java.util.Set;


/**
 * Abstract ImageSearcher, which uses javax.imageio.ImageIO to create a BufferedImage
 * from an InputStream.
 * <p/>
 * This file is part of the Caliph and Emir project: http://www.SemanticMetadata.net
 * <br>Date: 01.02.2006
 * <br>Time: 00:13:16
 *
 * @author Mathias Lux, mathias@juggle.at
 */
public abstract class AbstractImageSearcher implements ImageSearcher {
    /**
     * Searches for images similar to the given image. This simple implementation uses
     * {@link ImageSearcher#search(java.awt.image.BufferedImage, org.apache.lucene.index.IndexReader)},
     * the image is read using javax.imageio.ImageIO.
     *
     * @param image  the example image to search for.
     * @param reader the IndexReader which is used to dsearch through the images.
     * @return a sorted list of hits.
     * @throws IOException in case the image could not be read from stream.
     */
    public ImageSearchHits search(InputStream image, IndexReader reader) throws IOException {
        BufferedImage bufferedImage = ImageIO.read(image);
        return search(bufferedImage, reader);
    }

    public ImageSearchHits relevanceFeedback(ImageSearchHits originalSearch, Set<Document> positives, Set<Document> negatives) {
        throw new UnsupportedOperationException("Not implemented yet for this kind of searcher!");
    }
}
```

从代码中可以看出AbstractImageSearcher实现了ImageSearcher接口。其中的search(InputStream image, IndexReader reader)方法调用了search(BufferedImage image, IndexReader reader)方法。说白了，就是把2个函数的功能合并为一个函数。

ImageSearcherFactory

[java]

```java
/*
 * This file is part of the LIRe project: http://www.semanticmetadata.net/lire
 * LIRe is free software; you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation; either version 2 of the License, or
 * (at your option) any later version.
 *
 * LIRe is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with LIRe; if not, write to the Free Software
 * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA
 *
 * We kindly ask you to refer the following paper in any publication mentioning Lire:
 *
 * Lux Mathias, Savvas A. Chatzichristofis. Lire: Lucene Image Retrieval 鈥�
 * An Extensible Java CBIR Library. In proceedings of the 16th ACM International
 * Conference on Multimedia, pp. 1085-1088, Vancouver, Canada, 2008
 *
 * http://doi.acm.org/10.1145/1459359.1459577
 *
 * Copyright statement:
 * ~~~~~~~~~~~~~~~~~~~~
 * (c) 2002-2011 by Mathias Lux (mathias@juggle.at)
 *     http://www.semanticmetadata.net/lire
 */

package net.semanticmetadata.lire;

import net.semanticmetadata.lire.imageanalysis.*;
import net.semanticmetadata.lire.impl.CorrelogramImageSearcher;
import net.semanticmetadata.lire.impl.GenericFastImageSearcher;
import net.semanticmetadata.lire.impl.SimpleImageSearcher;

/**
 * <h2>Searching in an Index</h2>
 * Use the ImageSearcherFactory for creating an ImageSearcher, which will retrieve the images
 * for you from the index.
 * <p/>
 * <pre>
 * IndexReader reader = IndexReader.open(indexPath);
 * ImageSearcher searcher = ImageSearcherFactory.createDefaultSearcher();
 * FileInputStream imageStream = new FileInputStream("image.jpg");
 * BufferedImage bimg = ImageIO.read(imageStream);
 * // searching for an image:
 * ImageSearchHits hits = null;
 * hits = searcher.search(bimg, reader);
 * for (int i = 0; i < 5; i++) {
 * System.out.println(hits.score(i) + ": " + hits.doc(i).getField(DocumentBuilder.FIELD_NAME_IDENTIFIER).stringValue());
 * }
 *
 * // searching for a document:
 * Document document = hits.doc(0);
 * hits = searcher.search(document, reader);
 * for (int i = 0; i < 5; i++) {
 * System.out.println(hits.score(i) + ": " + hits.doc(i).getField(DocumentBuilder.FIELD_NAME_IDENTIFIER).stringValue());
 * }
 * </pre>
 * <p/>
 * This file is part of the Caliph and Emir project: http://www.SemanticMetadata.net
 * <br>Date: 03.02.2006
 * <br>Time: 00:30:07
 *
 * @author Mathias Lux, mathias@juggle.at
 */
public class ImageSearcherFactory {
    /**
     * Default number of maximum hits.
     */
    public static int NUM_MAX_HITS = 100;

    /**
     * Creates a new simple image searcher with the desired number of maximum hits.
     *
     * @param maximumHits
     * @return the searcher instance
     * @deprecated Use ColorLayout, EdgeHistogram and ScalableColor features instead.
     */
    public static ImageSearcher createSimpleSearcher(int maximumHits) {
        return ImageSearcherFactory.createColorLayoutImageSearcher(maximumHits);
    }

    /**
     * Returns a new default ImageSearcher with a predefined number of maximum
     * hits defined in the {@link ImageSearcherFactory#NUM_MAX_HITS} based on the {@link net.semanticmetadata.lire.imageanalysis.CEDD
```

```java
eature
     *
     * @return the searcher instance
     */
    public static ImageSearcher createDefaultSearcher() {
        return new GenericFastImageSearcher(NUM_MAX_HITS, CEDD.class, DocumentBuilder.FIELD_NAME_CEDD);
    }

    /**
     * Returns a new ImageSearcher with the given number of maximum hits
     * which only takes the overall color into account. texture and color
     * distribution are ignored.
     *
     * @param maximumHits defining how many hits are returned in max (e.g. 100 would be ok)
     * @return the ImageSearcher
     * @see ImageSearcher
     * @deprecated Use ColorHistogram or ScalableColor instead
     */
    public static ImageSearcher createColorOnlySearcher(int maximumHits) {
        return ImageSearcherFactory.createScalableColorImageSearcher(maximumHits);
    }

    /**
     * Returns a new ImageSearcher with the given number of maximum hits and
     * the specified weights on the different matching aspects. All weights
     * should be in [0,1] whereas a weight of 0 implies that the feature is
     * not taken into account for searching. Note that the effect is relative and
     * can only be fully applied if the {@link DocumentBuilderFactory#getExtensiveDocumentBuilder() extensive DocumentBuilder}
     * is used.
     *
     * @param maximumHits             defining how many hits are returned in max
     * @param colorHistogramWeight    a weight in [0,1] defining the importance of overall color in the images
     * @param colorDistributionWeight a weight in [0,1] defining the importance of color distribution (which color where) in the imag

     * @param textureWeight           defining the importance of texture (which edges where) in the images
     * @return the searcher instance or NULL if the weights are not appropriate, eg. all 0 or not in [0,1]
     * @see DocumentBuilderFactory
     * @deprecated Use ColorLayout, EdgeHistogram and ScalableColor features instead.
     */
    public static ImageSearcher createWeightedSearcher(int maximumHits,
                                                       float colorHistogramWeight,
                                                       float colorDistributionWeight,
                                                       float textureWeight) {
        if (isAppropriateWeight(colorHistogramWeight)
                && isAppropriateWeight(colorDistributionWeight)
                && isAppropriateWeight(textureWeight)
                && (colorHistogramWeight + colorDistributionWeight + textureWeight > 0f))
            return new SimpleImageSearcher(maximumHits, colorHistogramWeight, colorDistributionWeight, textureWeight);
        else
            return null;
    }

    /**
     * Create and return an ImageSearcher for the {@link net.semanticmetadata.lire.imageanalysis.AutoColorCorrelogram}
     * image feature. Be sure to use the same options for the ImageSearcher as you used for the DocumentBuilder.
     *
     * @param maximumHits number of hits returned.
     * @return
     */
    public static ImageSearcher createAutoColorCorrelogramImageSearcher(int maximumHits) {
        return new GenericFastImageSearcher(maximumHits, AutoColorCorrelogram.class, DocumentBuilder.FIELD_NAME_AUTOCOLORCORRELOGRAM)
;
//        return new CorrelogramImageSearcher(maximumHits, AutoColorCorrelogram.Mode.SuperFast);
    }

    /**
     * Create and return an ImageSearcher for the {@link net.semanticmetadata.lire.imageanalysis.AutoColorCorrelogram}
     * image feature. Be sure to use the same options for the ImageSearcher as you used for the DocumentBuilder.
     *
     * @param maximumHits number of hits returned.
     * @return
     * @deprecated Use #createAutoColorCorrelogramImageSearcher instead
     */
    public static ImageSearcher createFastCorrelogramImageSearcher(int maximumHits) {
        return new CorrelogramImageSearcher(maximumHits, AutoColorCorrelogram.Mode.SuperFast);
    }

    /**
     * Create and return an ImageSearcher for the {@link net.semanticmetadata.lire.imageanalysis.CEDD}
     * image feature. Be sure to use the same options for the ImageSearcher as you used for the DocumentBuilder.
     *
     * @param maximumHits
     * @return
     */
    public static ImageSearcher createCEDDImageSearcher(int maximumHits) {
//        return new CEDDImageSearcher(maximumHits);
        return new GenericFastImageSearcher(maximumHits, CEDD.class, DocumentBuilder.FIELD_NAME_CEDD);
    }


    /**
     * Create and return an ImageSearcher for the {@link net.semanticmetadata.lire.imageanalysis.FCTH}
     * Create and return an ImageSearcher for the {@link net.semanticmetadata.lire.imageanalysis.FCTH}
     * image feature. Be sure to use the same options for the ImageSearcher as you used for the DocumentBuilder.
     *
     * @param maximumHits
     * @return
     */
    public static ImageSearcher createFCTHImageSearcher(int maximumHits) {
//          return new GenericImageSearcher(maximumHits, FCTH.class, DocumentBuilder.FIELD_NAME_FCTH);
        return new GenericFastImageSearcher(maximumHits, FCTH.class, DocumentBuilder.FIELD_NAME_FCTH);
    }


    /**
     * Create and return an ImageSearcher for the {@link net.semanticmetadata.lire.imageanalysis.JCD}
     * image feature. Be sure to use the same options for the ImageSearcher as you used for the DocumentBuilder.
     *
     * @param maximumHits
     * @return
     */
    public static ImageSearcher createJCDImageSearcher(int maximumHits) {
        return new GenericFastImageSearcher(maximumHits, JCD.class, DocumentBuilder.FIELD_NAME_JCD);
    }


    /**
     * Create and return an ImageSearcher for the {@link net.semanticmetadata.lire.imageanalysis.JpegCoefficientHistogram}
     * image feature. Be sure to use the same options for the ImageSearcher as you used for the DocumentBuilder.
     *
     * @param maximumHits
     * @return
     */
    public static ImageSearcher createJpegCoefficientHistogramImageSearcher(int maximumHits) {
        return new GenericFastImageSearcher(maximumHits, JpegCoefficientHistogram.class, DocumentBuilder.FIELD_NAME_JPEGCOEFFS);
    }


    /**
     * Create and return an ImageSearcher for the {@link net.semanticmetadata.lire.imageanalysis.SimpleColorHistogram}
     * image feature. Be sure to use the same options for the ImageSearcher as you used for the DocumentBuilder.
     *
     * @param maximumHits
     * @return
     */
    public static ImageSearcher createColorHistogramImageSearcher(int maximumHits) {
//          return new GenericImageSearcher(maximumHits, SimpleColorHistogram.class, DocumentBuilder.FIELD_NAME_COLORHISTOGRAM);
        return new GenericFastImageSearcher(maximumHits, SimpleColorHistogram.class, DocumentBuilder.FIELD_NAME_COLORHISTOGRAM);
    }

    /**
     * Create and return an ImageSearcher for the {@link net.semanticmetadata.lire.imageanalysis.Tamura}
     * image feature. Be sure to use the same options for the ImageSearcher as you used for the DocumentBuilder.
     *
     * @param maximumHits
     * @return
     */
    public static ImageSearcher createTamuraImageSearcher(int maximumHits) {
        return new GenericFastImageSearcher(maximumHits, Tamura.class, DocumentBuilder.FIELD_NAME_TAMURA);
    }

    /**
     * Create and return an ImageSearcher for the {@link net.semanticmetadata.lire.imageanalysis.Gabor}
     * image feature. Be sure to use the same options for the ImageSearcher as you used for the DocumentBuilder.
     *
     * @param maximumHits
     * @return
     */
    public static ImageSearcher createGaborImageSearcher(int maximumHits) {
        return new GenericFastImageSearcher(maximumHits, Gabor.class, DocumentBuilder.FIELD_NAME_GABOR);
    }

    /**
     * Create and return an ImageSearcher for the {@link net.semanticmetadata.lire.imageanalysis.ColorLayout}
     * image feature using the byte[] serialization. Be sure to use the same options for the ImageSearcher as
     * you used for the DocumentBuilder.
     *
     * @param maximumHits
     * @return
     */
    public static ImageSearcher createColorLayoutImageSearcher(int maximumHits) {
        return new GenericFastImageSearcher(maximumHits, ColorLayout.class, DocumentBuilder.FIELD_NAME_COLORLAYOUT);
    }

    /**
     * Create and return an ImageSearcher for the {@link net.semanticmetadata.lire.imageanalysis.ScalableColor}
     * image feature using the byte[] serialization. Be sure to use the same options for the ImageSearcher as
     * you used for the DocumentBuilder.
     *
     * @param maximumHits
     * @return
     */
    public static ImageSearcher createScalableColorImageSearcher(int maximumHits) {
        return new GenericFastImageSearcher(maximumHits, ScalableColor.class, DocumentBuilder.FIELD_NAME_SCALABLECOLOR);
    }

    /**
     * Create and return an ImageSearcher for the {@link net.semanticmetadata.lire.imageanalysis.EdgeHistogram}
     * image feature using the byte[] serialization. Be sure to use the same options for the ImageSearcher as
     * you used for the DocumentBuilder.
     *
     * @param maximumHits
     * @return
     */
    public static ImageSearcher createEdgeHistogramImageSearcher(int maximumHits) {
        return new GenericFastImageSearcher(maximumHits, EdgeHistogram.class, DocumentBuilder.FIELD_NAME_EDGEHISTOGRAM);
    }


    /**
     * Checks if the weight is in [0,1]
     *
     * @param f the weight to check
     * @return true if the weight is in [0,1], false otherwise
     */
    private static boolean isAppropriateWeight(float f) {
        boolean result = false;
        if (f <= 1f && f >= 0) result = true;
        return result;

    }
}
```

ImageSearcherFactory是用于创建ImageSearcher的。里面有各种create****ImageSearcher()。每个函数的作用在注释中都有详细的说明。

ImageSearchHits

[java]

```java
/*
 * This file is part of the LIRe project: http://www.semanticmetadata.net/lire
 * LIRe is free software; you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation; either version 2 of the License, or
 * (at your option) any later version.
 *
 * LIRe is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with LIRe; if not, write to the Free Software
 * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA
 *
 * We kindly ask you to refer the following paper in any publication mentioning Lire:
 *
 * Lux Mathias, Savvas A. Chatzichristofis. Lire: Lucene Image Retrieval 鈥�
 * An Extensible Java CBIR Library. In proceedings of the 16th ACM International
 * Conference on Multimedia, pp. 1085-1088, Vancouver, Canada, 2008
 *
 * http://doi.acm.org/10.1145/1459359.1459577
 *
 * Copyright statement:
 * --------------------
 * (c) 2002-2011 by Mathias Lux (mathias@juggle.at)
 *     http://www.semanticmetadata.net/lire
 */

package net.semanticmetadata.lire;

import org.apache.lucene.document.Document;

/**
 * This class simulates the original Lucene Hits object.
 * Please note the only a certain number of results are returned.<br>
 * <p/>
 * This file is part of the Caliph and Emir project: http://www.SemanticMetadata.net
 * <br>Date: 02.02.2006
 * <br>Time: 23:45:20
 *
 * @author Mathias Lux, mathias@juggle.at
 */
public interface ImageSearchHits {
    /**
     * Returns the size of the result list.
     *
     * @return the size of the result list.
     */
    public int length();

    /**
     * Returns the score of the document at given position.
     * Please note that the score in this case is a distance,
     * which means a score of 0 denotes the best possible hit.
     * The result list starts with position 0 as everything
     * in computer science does.
     *
     * @param position defines the position
     * @return the score of the document at given position. The lower the better (its a distance measure).
     */
    public float score(int position);

    /**
     * Returns the document at given position
     *
     * @param position defines the position.
     * @return the document at given position.
     */
    public Document doc(int position);
}
```

该类主要用于存储ImageSearcher类中search()方法返回的结果。

SimpleImageSearchHits是ImageSearcher的实现。该类的源代码如下所示：

[java]

```java
/*
 * This file is part of the LIRe project: http://www.semanticmetadata.net/lire
 * LIRe is free software; you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation; either version 2 of the License, or
 * (at your option) any later version.
 *
 * LIRe is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with LIRe; if not, write to the Free Software
 * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA
 *
 * We kindly ask you to refer the following paper in any publication mentioning Lire:
 *
 * Lux Mathias, Savvas A. Chatzichristofis. Lire: Lucene Image Retrieval 鈥�
 * An Extensible Java CBIR Library. In proceedings of the 16th ACM International
 * Conference on Multimedia, pp. 1085-1088, Vancouver, Canada, 2008
 *
 * http://doi.acm.org/10.1145/1459359.1459577
 *
 * Copyright statement:
 * --------------------
 * (c) 2002-2011 by Mathias Lux (mathias@juggle.at)
 *     http://www.semanticmetadata.net/lire
 */

package net.semanticmetadata.lire.impl;

import net.semanticmetadata.lire.ImageSearchHits;
import org.apache.lucene.document.Document;

import java.util.ArrayList;
import java.util.Collection;
import java.util.Iterator;

/**
 * This file is part of the Caliph and Emir project: http://www.SemanticMetadata.net
 * <br>Date: 02.02.2006
 * <br>Time: 23:56:15
 *
 * @author Mathias Lux, mathias@juggle.at
 */
public class SimpleImageSearchHits implements ImageSearchHits {
    ArrayList<SimpleResult> results;

    public SimpleImageSearchHits(Collection<SimpleResult> results, float maxDistance) {
        this.results = new ArrayList<SimpleResult>(results.size());
        this.results.addAll(results);
        // this step normalizes and inverts the distance ...
        // although its now a score or similarity like measure its further called distance
        for (Iterator<SimpleResult> iterator = this.results.iterator(); iterator.hasNext(); ) {
            SimpleResult result = iterator.next();
            result.setDistance(1f - result.getDistance() / maxDistance);
        }
    }

    /**
     * Returns the size of the result list.
     *
     * @return the size of the result list.
     */
    public int length() {
        return results.size();
    }

    /**
     * Returns the score of the document at given position.
     * Please note that the score in this case is a distance,
     * which means a score of 0 denotes the best possible hit.
     * The result list starts with position 0 as everything
     * in computer science does.
     *
     * @param position defines the position
     * @return the score of the document at given position. The lower the better (its a distance measure).
     */
    public float score(int position) {
        return results.get(position).getDistance();
    }

    /**
     * Returns the document at given position
     *
     * @param position defines the position.
     * @return the document at given position.
     */
    public Document doc(int position) {
        return results.get(position).getDocument();
    }

    private float sigmoid(float f) {
        double result = 0f;
        result = -1d + 2d / (1d + Math.exp(-2d * f / 0.6));
        return (float) (1d - result);
    }
}
```

可以看出检索的结果是存在名为results的ArrayList<SimpleResult> 类型的变量中的。

文章标签： lire 源代码 索引 检索 lucene

个人分类： MPEG7/图像检索 LIRe

所属专栏： 开源多媒体项目源代码分析