

盘古分词使用手册

文件说明	2
PanGuSegment	2
PanGu4Lucene.....	2
PanGu.dll 调用方法	2
初始化	2
分词	2
配置文件 PanGu.xml.....	7
高亮组件 PanGu.HighLight.dll 调用方法.....	8
字典管理	8
Demo.exe.....	11
PanGu4Lucene 调用方法	12
创建索引.....	12
插入数据.....	12
对要搜索的词分词	13
搜索	13
PanGu4Lucene 示例	15
PanGu4Lucene 示例安装说明.....	15

文件说明

PanGuSegment

这个是盘古分词的组件包，包括
PanGu.dll 盘古分词的核心组件
DictManage.exe 字典管理工具
Demo.exe 分词演示程序
PanGu.HighLight.dll 高亮组件

PanGu4Lucene

这个是盘古分词针对 Lucene.net 提供的接口
PanGu.Lucene.Analyzer.dll 盘古分词针对 Lucene.net 的接口组件
PanGu.Lucene.ImportTool.exe 示例程序数据导入程序

PanGu.dll 调用方法

初始化

在进程启动时，我们需要对盘古分词进行初始化，初始化的调用代码如下：

```
PanGu.Segment.Init();  
或  
PanGu.Segment.Init(filename);
```

filename 为 pangu.xml 的完整路径名，如 “c:\pangu.xml”

分词

```
Segment segment = new Segment();  
ICollection<WordInfo> words = segment.DoSegment(text);  
或  
ICollection<WordInfo> words = segment.DoSegment(text, options);  
或  
ICollection<WordInfo> words = segment.DoSegment(text, options, parameters);
```

其中

- text 为需要分词的文本
- options 为自定义分词选项，默认为 pangu.xml 中指定的分词选项
- parameters 为分词参数，默认为 pangu.xml 中指定的分词参数

分词选项定义：

```
public class MatchOptions
{
    /// <summary>
    /// 中文人名识别
    /// </summary>
    public bool ChineseNameIdentify = false;

    /// <summary>
    /// 词频优先
    /// </summary>
    public bool FrequencyFirst = false;

    /// <summary>
    /// 多元分词
    /// </summary>
    public bool MultiDimensionality = true;

    /// <summary>
    /// 过滤停用词
    /// </summary>
    public bool FilterStopWords = true;

    /// <summary>
    /// 忽略空格、回车、Tab
    /// </summary>
    public bool IgnoreSpace = true;

    /// <summary>
    /// 强制一元分词
    /// </summary>
    public bool ForceSingleWord = false;

    /// <summary>
    /// 繁体中文开关
    /// </summary>
    public bool TraditionalChineseEnabled = false;

    /// <summary>
    /// 同时输出简体和繁体
    /// </summary>
    public bool OutputSimplifiedTraditional = false;

    /// <summary>
    /// 未登录词识别
    /// </summary>
    public bool UnknownWordIdentify = true;

    /// <summary>
    /// 过滤英文，这个选项只有在过滤停用词选项生效时才有效
    /// </summary>
    public bool FilterEnglish = false;

    /// <summary>
    /// 过滤数字，这个选项只有在过滤停用词选项生效时才有效
    /// </summary>
```

```

public bool FilterNumeric = false;

/// <summary>
/// 忽略英文大小写
/// </summary>
public bool IgnoreCapital = false;

/// <summary>
/// 英文分词
/// </summary>
public bool EnglishSegment = false;

/// <summary>
/// 同义词输出
/// </summary>
/// <remarks>
/// 同义词输出功能一般用于对搜索字符串的分词，不建议在索引时使用
/// </remarks>
public bool SynonymOutput = false;

/// <summary>
/// 通配符匹配输出
/// </summary>
/// <remarks>
/// 同义词输出功能一般用于对搜索字符串的分词，不建议在索引时使用
/// </remarks>
public bool WildcardOutput = false;

/// <summary>
/// 对通配符匹配的结果分词
/// </summary>
public bool WildcardSegment = false;

/// <summary>
/// 是否进行用户自定义规则匹配
/// </summary>
public bool CustomRule = false;

}

```

分词参数定义

```

[Serializable]
public class MatchParameter
{

```

```
/// <summary>
/// 多元分词冗余度
/// </summary>
public int Redundancy = 0;

/// <summary>
/// 未登录词权值
/// </summary>
public int UnknowRank = 1;

/// <summary>
/// 最匹配词权值
/// </summary>
public int BestRank = 5;

/// <summary>
/// 次匹配词权值
/// </summary>
public int SecRank = 3;

/// <summary>
/// 再次匹配词权值
/// </summary>
public int ThirdRank = 2;

/// <summary>
/// 强行输出的单字的权值
/// </summary>
public int SingleRank = 1;

/// <summary>
/// 数字的权值
/// </summary>
public int NumericRank = 1;

/// <summary>
/// 英文词汇权值
/// </summary>
public int EnglishRank = 5;

/// <summary>
/// 符号的权值
/// </summary>
public int SymbolRank = 1;

/// <summary>
/// 强制同时输出简繁汉字时，非原来文本的汉字输出权值。
/// 比如原来文本是简体，这里就是输出的繁体字的权值，反之亦然。
/// </summary>
public int SimplifiedTraditionalRank = 1;

/// <summary>
/// 同义词权值
/// </summary>
public int SynonymRank = 1;
```

```

    /// <summary>
    /// 通配符匹配结果的权值
    /// </summary>
    public int WildcardRank = 1;

    /// <summary>
    /// 过滤英文选项生效时，过滤大于这个长度的英文。
    /// </summary>
    public int FilterEnglishLength = 0;

    /// <summary>
    /// 过滤数字选项生效时，过滤大于这个长度的数字。
    /// </summary>
    public int FilterNumericLength = 0;

    /// <summary>
    /// 用户自定义规则的配件文件名
    /// </summary>
    public string CustomRuleAssemblyFileName = "";

    /// <summary>
    /// 用户自定义规则的类的完整名，即带名字空间的名称
    /// </summary>
    public string CustomRuleFullClassName = "";

}

返回为 WordInfo 的集合
WordInfo 定义

```

```

public class WordInfo : WordAttribute, IComparable<WordInfo>
{
    /// <summary>
    /// 当前单词类型
    /// </summary>
    public WordType WordType;

    /// <summary>
    /// 原始的单词类型
    /// </summary>
    public WordType OriginalWordType;

    /// <summary>
    /// 单词在text 中的起始位置
    /// </summary>
    public int Position;

    /// <summary>
    /// Rank for this word
    /// 单词权重
    /// </summary>
    public int Rank;

    /// <summary>

```

```

    /// 单词
    /// </summary>
    public String Word;

    /// <summary>
    /// 词性
    /// </summary>
    public POS Pos;

    /// <summary>
    /// 词频
    /// </summary>
    public double Frequency;
}

```

配置文件 PanGu.xml

```

<?xml version="1.0" encoding="utf-8"?>
<PanGuSettings xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns="http://www.codeplex.com/pangusegment">
    <DictionaryPath>..\Dictionaries</DictionaryPath>
    <MatchOptions>
        <ChineseNameIdentify>true</ChineseNameIdentify>
        <FrequencyFirst>false</FrequencyFirst>
        <MultiDimensionality>false</MultiDimensionality>
        <FilterStopWords>true</FilterStopWords>
        <IgnoreSpace>true</IgnoreSpace>
        <ForceSingleWord>false</ForceSingleWord>
        <TraditionalChineseEnabled>false</TraditionalChineseEnabled>
        <OutputSimplifiedTraditional>false</OutputSimplifiedTraditional>
        <UnknownWordIdentify>true</UnknownWordIdentify>
        <FilterEnglish>false</FilterEnglish>
        <FilterNumeric>false</FilterNumeric>
        <IgnoreCapital>false</IgnoreCapital>
        <EnglishSegment>false</EnglishSegment>
        <SynonymOutput>false</SynonymOutput>
        <WildcardOutput>false</WildcardOutput>
        <WildcardSegment>false</WildcardSegment>
        <CustomRule>false</CustomRule>
    </MatchOptions>
    <Parameters>
        <UnknowRank>1</UnknowRank>
    
```

```

<BestRank>5</BestRank>
<SecRank>3</SecRank>
<ThirdRank>2</ThirdRank>
<SingleRank>1</SingleRank>
<NumericRank>1</NumericRank>
<EnglishRank>5</EnglishRank>
<EnglishLowerRank>3</EnglishLowerRank>
<EnglishStemRank>2</EnglishStemRank>
<SymbolRank>1</SymbolRank>
<SimplifiedTraditionalRank>1</SimplifiedTraditionalRank>
<SynonymRank>1</SynonymRank>
<WildcardRank>1</WildcardRank>
<FilterEnglishLength>0</FilterEnglishLength>
<FilterNumericLength>0</FilterNumericLength>

<CustomRuleAssemblyFileName>CustomRuleExample.dll</CustomRuleAssemblyFileName>

<CustomRuleFullClassName>CustomRuleExample.PickupNokia</CustomRuleFullClassName>
    <Redundancy>0</Redundancy>
</Parameters>
</PanGuSettings>

```

其中 DictionaryPath 指明字典所在目录，可以为相对路径也可以为绝对路径。

MatchOptions 对应分词选项

Parameters 对于分词参数

高亮组件 PanGu.HighLight.dll 调用方法

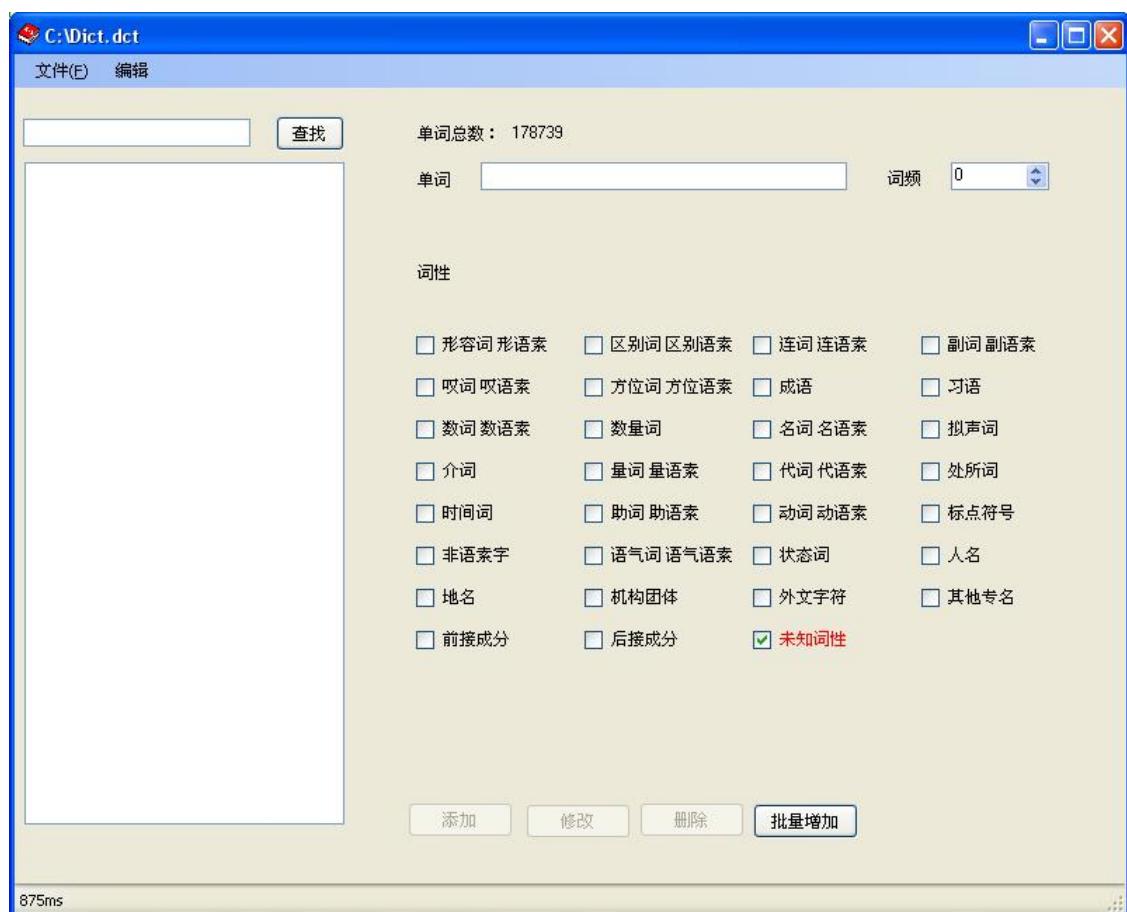
```

//创建HTMLFormatter,参数为高亮单词的前后缀
PanGu.HighLight.SimpleHTMLFormatter simpleHTMLFormatter =
    new PanGu.HighLight.SimpleHTMLFormatter("<font color=\"red\">", "</font>");
//创建 Highlighter , 输入HTMLFormatter 和 盘古分词对象Segment
PanGu.HighLight.HighLighter highlighter =
    new PanGu.HighLight.HighLighter(simpleHTMLFormatter,
    new Segment());
//设置每个摘要段的字符数
highlighter.FragmentSize = 50;
//获取最匹配的摘要段
String abstract = highlighter.GetBestFragment(keywords, news.Content);

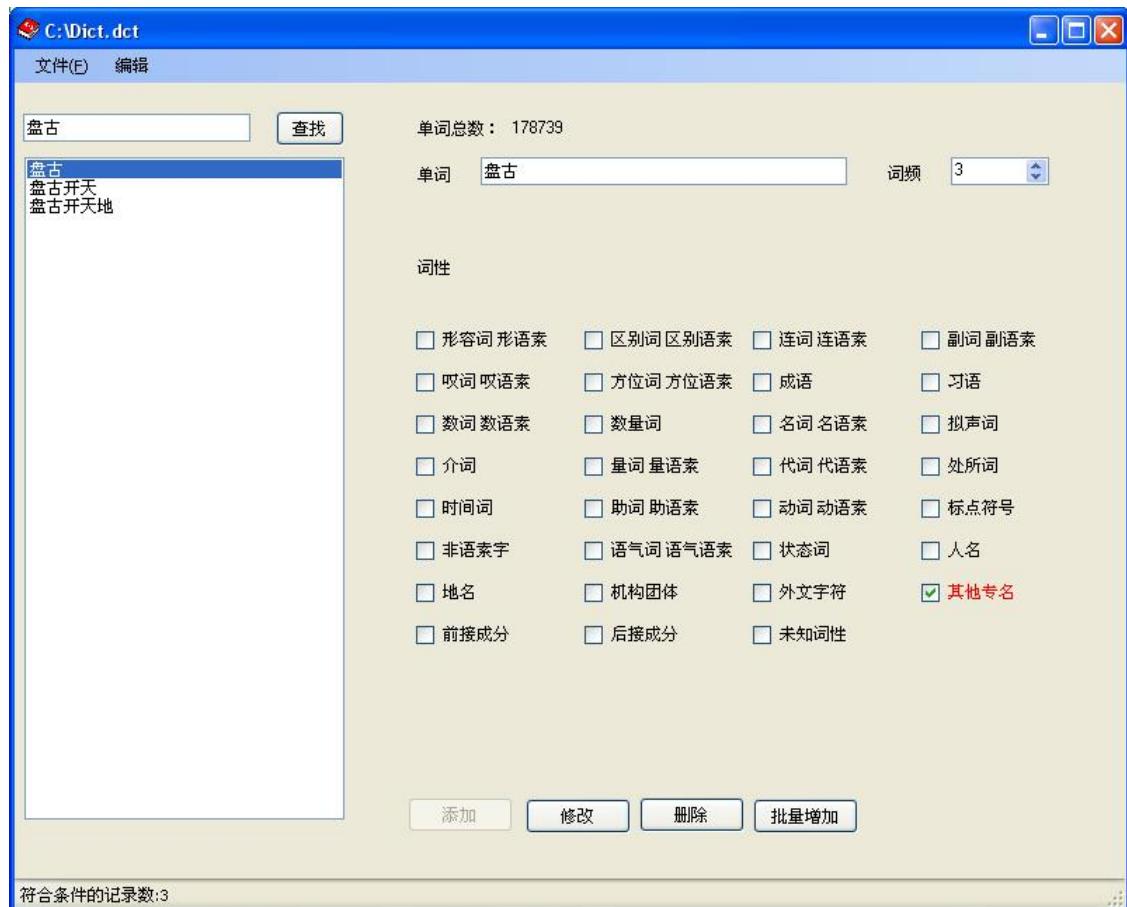
```

字典管理

首先在文件菜单中点打开，然后打开字典文件 dict.dct

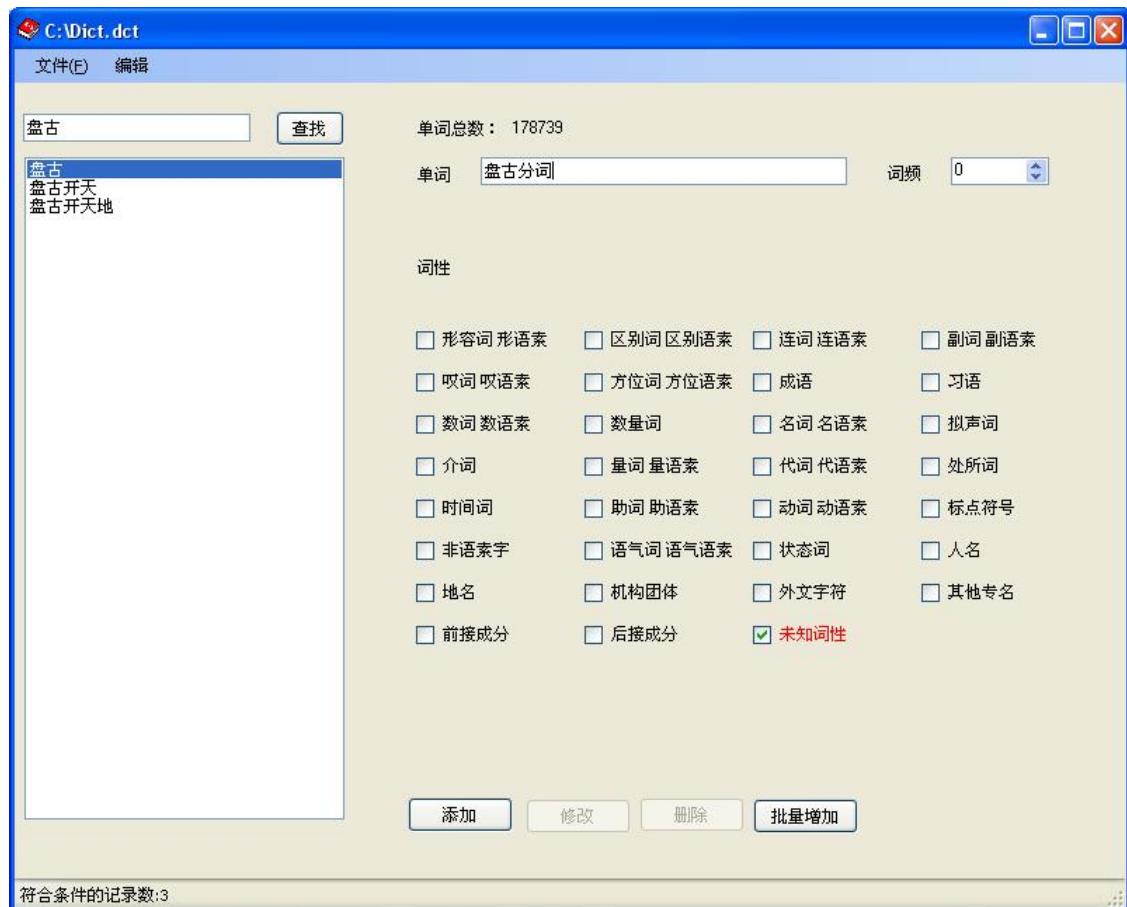


在文本框中输入单词可以查找字典中包含这个单词的所有词



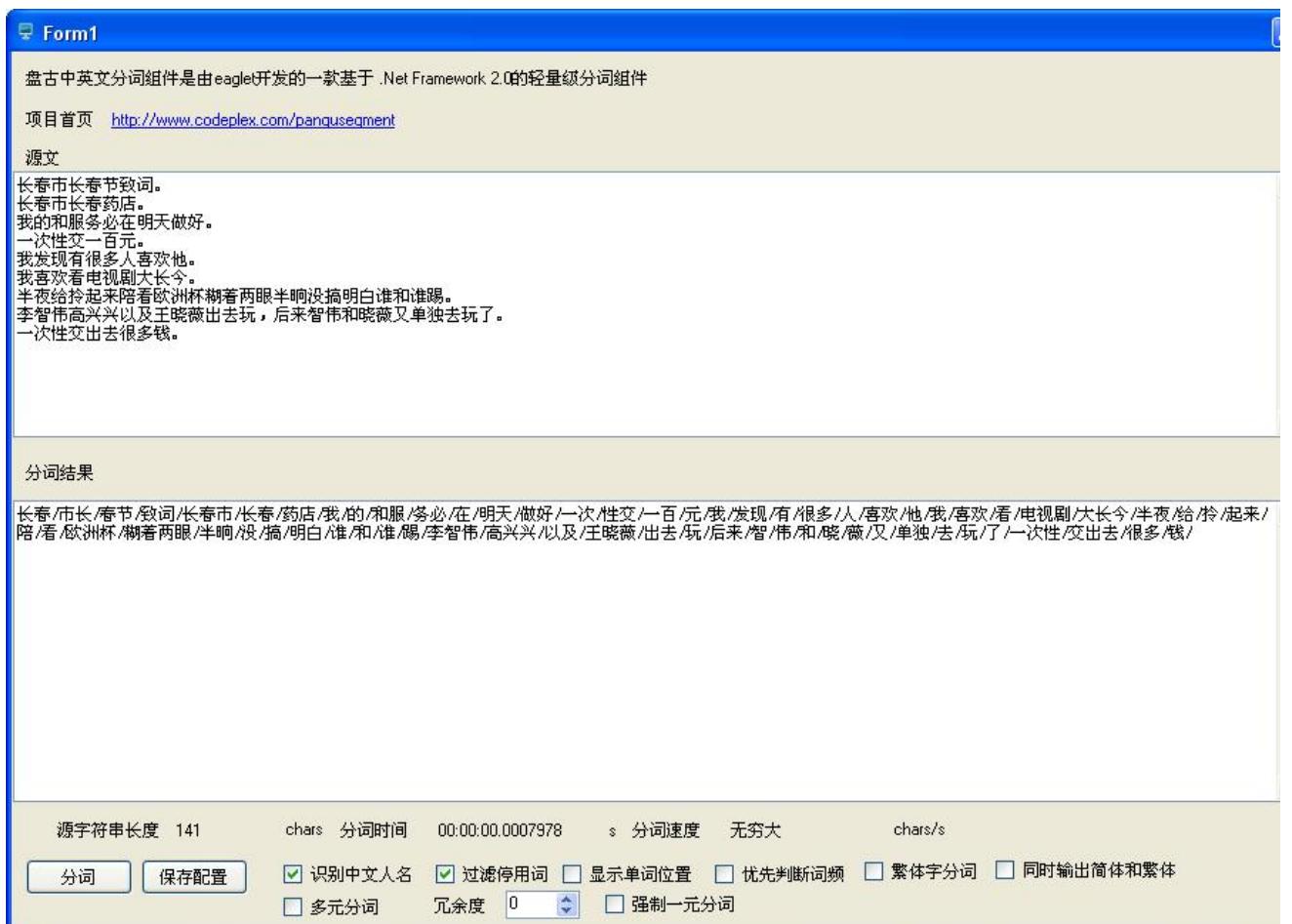
这时你可以修改或删除已有单词

如果要添加新单词，可以在单词后面的输入框输入新单词



然后点击添加就可以了。

Demo.exe



PanGu4Lucene 调用方法

创建索引

```
IndexWriter writer = new IndexWriter(indexDir, new PanGuAnalyzer(), true);
writer.Optimize();
writer.Close();
```

插入数据

```
public static int IndexString(String indexDir, string url, string title, DateTime time,
string content)
{
    //IndexWriter writer = new IndexWriter(indexDir, new
Lucene.Net.Analysis.KTDictSeg.KTDictSegAnalyzer(), false);
```

```

Document doc = new Document();

Field field = new Field("url", url, Field.Store.YES, Field.Index.NO);
doc.Add(field);
field = new Field("title", title, Field.Store.YES, Field.Index.TOKENIZED);
doc.Add(field);
field = new Field("time", time.ToString("yyyyMMdd"), Field.Store.YES,
Field.Index.UN_TOKENIZED);
doc.Add(field);
field = new Field("contents", content, Field.Store.YES, Field.Index.TOKENIZED);
doc.Add(field);

writer.AddDocument(doc);

int num = writer.DocCount();
//writer.Optimize();
//writer.Close();
return num;
}

```

对要搜索的词分词

```

static public string GetKeyWordsSplitBySpace(string keywords, PanGuTokenizer ktTokenizer)
{
    StringBuilder result = new StringBuilder();

    ICollection<WordInfo> words = ktTokenizer.SegmentToWordInfos(keywords);

    foreach (WordInfo word in words)
    {
        if (word == null)
        {
            continue;
        }

        result.AppendFormat("{0}^{1}.0 ", word.Word, (int)Math.Pow(3, word.Rank));
    }

    return result.ToString().Trim();
}

```

搜索

```

public static List<TNews> Search(String indexDir, String q, int pageLen, int pageNo, out
int recCount)
{
    string keywords = q;

    IndexSearcher search = new IndexSearcher(indexDir);
    q = GetKeyWordsSplitBySpace(q, new PanGuTokenizer());
    QueryParser queryParser = new QueryParser("contents", new PanGuAnalyzer(true));
    Query query = queryParser.Parse(q);

```

```

QueryParser titleQueryParser = new QueryParser("title", new PanGuAnalyzer(true));
Query titleQuery = titleQueryParser.Parse(q);

BooleanQuery bq = new BooleanQuery();
bq.Add(query, BooleanClause.Occur.SHOULD);
bq.Add(titleQuery, BooleanClause.Occur.SHOULD);

Hits hits = search.Search(bq);

List<TNews> result = new List<TNews>();

recCount = hits.Length();
int i = (pageNo - 1) * pageLen;

while (i < recCount && result.Count < pageLen)
{
    TNews news = null;

    try
    {
        news = new TNews();
        news.Title = hits.Doc(i).Get("title");
        news.Content = hits.Doc(i).Get("contents");
        news.Url = hits.Doc(i).Get("url");
        String strTime = hits.Doc(i).Get("time");
        news.Time = DateTime.ParseExact(strTime, "yyyyMMdd", null);

        PanGu.HighLight.SimpleHTMLFormatter simpleHTMLFormatter =
            new PanGu.HighLight.SimpleHTMLFormatter("<font color=\"red\">",
            "</font>");

        PanGu.HighLight.HighLighter highlighter =
            new PanGu.HighLight.HighLighter(simpleHTMLFormatter,
            new Segment());

        highlighter.FragmentSize = 50;

        news.Abstract = highlighter.GetBestFragment(keywords, news.Content);
        news.TitleHighlighter = highlighter.GetBestFragment(keywords, news.Title);
        if (string.IsNullOrEmpty(news.TitleHighlighter))
        {
            news.TitleHighlighter = news.Title;
        }
    }
    catch (Exception e)
    {
        Console.WriteLine(e.Message);
    }
    finally
    {
        result.Add(news);
        i++;
    }
}

```

```
        }

        search.Close();
        return result;
    }
}
```

PanGu4Lucene 示例



PanGu4Lucene 示例安装说明

1. 下载 News.xml 下载地址
<http://pangusegment.codeplex.com/Release/ProjectReleases.aspx?ReleaseId=31482>
2. 进入目录 Bin, 并运行 PanGu.Lucene.ImportTool.exe 点击创建索引按钮, 并导入 news.xml
3. 运行网站