http://mail-archives.apache.org/mod\_mbox/lucene-dev/200110.mbox/%3c3BCF41C9.6080306@earthlink.net%3e

From lucene-dev-return-252-apmail-jakarta-lucene-dev-archive=jakarta.apache.org@jakarta.apache.org Thu Oct 18 20:55:45 2001

Return-Path: <lucene-dev-return-252-apmail-jakarta-lucene-dev-archive=jakarta.apache.org@jakarta.apache.org>

Delivered-To: apmail-jakarta-lucene-dev-archive@jakarta.apache.org

Received: (qmail 65250 invoked by uid 500); 18 Oct 2001 20:55:45 -0000

Mailing-List: contact lucene-dev-help@jakarta.apache.org; run by ezmlm

Precedence: bulk

Reply-To: lucene-dev@jakarta.apache.org

list-help: <mailto:lucene-dev-help@jakarta.apache.org>

list-unsubscribe: <mailto:lucene-dev-unsubscribe@jakarta.apache.org>

list-post: <mailto:lucene-dev@jakarta.apache.org>

Delivered-To: mailing list lucene-dev@jakarta.apache.org

Received: (qmail 64887 invoked from network); 18 Oct 2001 20:55:34 -0000

Message-ID: <3BCF41C9.6080306@earthlink.net>

Date: Thu, 18 Oct 2001 14:55:37 -0600

From: Dmitry Serebrennikov <dmitrys@earthlink.net>

User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.0; en-US; rv:0.9.2) Gecko/20010726 Netscape6/6.1

X-Accept-Language: en-us

MIME-Version: 1.0

To: lucene-dev@jakarta.apache.org

Subject: TermVector support - first release

Content-Type: multipart/mixed;

boundary="------------010909000801060505060408"

X-Spam-Rating: daedalus.apache.org 1.6.2 0/1000/N

--------------010909000801060505060408

Content-Type: text/plain; charset=us-ascii; format=flowed

Content-Transfer-Encoding: 7bit

Greetings, everyone!

I have the first version of the term vector support ready to go. I'm

attaching a file with release notes that explain breifly what the new

capabilities are and what there changes were to make the happen. There

are some limitations that are also described. The zip file contains new

files, to be added. The txt file is the result of cvs diff -u against

the current CVS repository.

I am really interested in feedback. First, do the APIs work for your

needs? Also, does everything work? What kind of performance you are

seeing? Are there things that could be done better (especially in terms

of file structures and reading of those files, I think this is where the

next layer of optimizations should come from).

In terms of riskiness, these changes are pretty risky, so I don't think

they should go into the 1.2. But I've been using them for the past few

days and I didn't have to touch the files at all, so I think they are

pretty stable.

Have fun, everyone.

Dmitry.

--------------010909000801060505060408

Content-Type: text/plain;

name="tv\_notes.txt"

Content-Transfer-Encoding: 7bit

Content-Disposition: inline;

filename="tv\_notes.txt"

Term Vector Support

Release Notes

===================

<<< WARNING >>>

This is only a preliminary release. It probably contains bugs. However, it does

work and it works well, at least in the tests that were conducted so far.

File formats are not final. They may change in the future, in which case indexes

created by this code may no longer be supported. If you use this code, you may need

to recreate your indexes in the future.

<<< New capabilities >>>

This set of changes adds support for storing and retrieving term vectors.

A term vector is composed of a set of unique terms that occur in a document's field

and frequencies with which they occur. In addition, a positions in which these terms

occur can also be retrieved.

API:

----

The following methods have been added to IndexReader:

TermFreqVector getTermFreqVector(int doc, String field);

TermFreqVector[] getTermFreqVectors(int doc);

TermPositionVector getTermPositionVector(int doc, String field);

TermPositionVector[] getTermPositionVectors(int doc);

Term getTerm(int termNumber);

Compatibility:

--------------

The index files are backward and forward compatible. That is old indexes can be

used with the new engine and new indexes with the old engine (not tested). Of course

term vectors are not available in old indexes running in the new engine. Likewise,

term vector information will be lost if the new index segments are merged with an

old engine.

The API changes are such that old applications should function without change

with the new engine. They will not incure the term vector overhead either.

(Except for some additional files created during segment merges).

Performance:

------------

As compared to storing this information in document fields, the new code is about

2 times faster just accessing the information! In addition, it refers to terms by

number rather than by their text and this allows any code that manipulates this

information to be much faster as well.

Unless term vector functionality is used, there is no query-time overhead incured.

There is some merge-time overhead, but not much. Also, there are new files that

add some storage overhead even if term vectors are not used.

Limitations:

------------

At this time, term vectors are only available in optimized indexes. If an index

contains more than one segment, attempts to call any of the term vector methods

will result in a RuntimeException. This should change in the future.

Also, there are probably additional ways in which the code can be optimized.

This should help even further reduce query times and maybe reduce the storage

requirements.

<<< TV support Changes from Lucene-1.2-rc1 >>>

- new files

src/java/org/apache/lucene/index/SegmentTermVector.java

src/java/org/apache/lucene/index/TermFreqVector.java

src/java/org/apache/lucene/index/TermPositionVector.java

src/java/org/apache/lucene/index/TermVectorsReader.java

src/java/org/apache/lucene/index/TermVectorsWriter.java

- changed src/java/org/apache/lucene/document/Field.java

Added a new attribute: "storeTermVector", which indicates that

field's term vector should be stored. This can only be set on

indexed fields. Added a new constructor and a few new static

methods versions that take this as an argument. All previously

existing constructors and methods set this attribute to false

to preserve compatibility.

- changed src/java/org/apache/lucene/index/DocumentWriter.java

Added support for extraction and storage of term vectors.

Biffedup stream closing code in the finally {} clause to attempt

to close as many streams as it can even if some streams throw

an exception during closing. The first exception caught is

preserved and rethrown after the streams are closed.

- changed src/java/org/apache/lucene/index/FieldInfo.java

Added the "storeTermVector" flag, changed constructor to take

a value for this flag.

- changed src/java/org/apache/lucene/index/FieldInfos.java

Added support for the "storeTermVector" flag.

Changed addInternal to return the number of the added field.

Changed add(FieldInfos) to return an array of field numbers, such

that it maps the field numbers in the argument FieldInfos to the

fields in this one (either newly added fields or previously

existing field). This is useful for term vector merging.

- changed src/java/org/apache/lucene/index/FieldsReader.java

Added a comment.

- changed src/java/org/apache/lucene/index/IndexReader.java

Added declarations for the term vector support methods.

- changed src/java/org/apache/lucene/index/SegmentMergeInfo.java

Added comments. Added a int termMap[] and the code to initialize it.

This map helps during term vector merge.

- changed src/java/org/apache/lucene/index/SegmentMergeQueue.java

Added comments.

- changed src/java/org/apache/lucene/index/SegmentMerger.java

Added comments. Added a method mergeVectors() and a few map

arrays that are used to merge term vector information. Changed

mergeTermInfos and appendPostions to maintain the term vector

and mapping info.

- changed src/java/org/apache/lucene/index/SegmentReader.java

Added new files to the list of segment files. Added support for

the new methods on IndexReader.

- changed src/java/org/apache/lucene/index/SegmentTermDocs.java

Added support for isValid() method.

- changed src/java/org/apache/lucene/index/SegmentTermEnum.java

Added support for isValid() and termNumber() methods.

- changed src/java/org/apache/lucene/index/SegmentTermPositions.java

Added a descriptive string to an Exception.

- changed src/java/org/apache/lucene/index/SegmentsReader.java

Added placeholders for IndexReader methods. Added support for

isValid and termNumber.

- changed src/java/org/apache/lucene/index/TermDocs.java

Added isValid method that indicates if there is a current document

in the enumeration.

- changed src/java/org/apache/lucene/index/TermEnum.java

Added isValid method that indicates if there is a current term

in the enumeration. Added termNumber method that returns a number

of a specified term. Added skipTo(Term) method that acts just like

TermDocs.skipTo.

- changed src/java/org/apache/lucene/index/TermInfosWriter.java

Added methods to return current position and total size.

- changed src/java/org/apache/lucene/search/IndexSearcher.java

Made maxDoc and docFreq to be public

- changed src/java/org/apache/lucene/search/MultiSearcher.java

Made maxDoc and docFreq to be public. Added support count() method.

- changed src/java/org/apache/lucene/search/Searcher.java

Added count() methods that return the number of documents

selected by a query without having to score the documents.

Made maxDoc and docFreq to be public.

<<< Other changes from Lucene-1.2-rc1 >>>

- changed src/java/org/apache/lucene/analysis/PorterStemmer.java

made the class public - this is unreleated to TV support

- changed src/java/org/apache/lucene/store/OutputStream.java

added comments

- changed src/java/org/apache/lucene/store/InputStream.java

added comments

--------------010909000801060505060408

Content-Type: application/x-zip-compressed;

name="tv\_newFiles.zip"

Content-Transfer-Encoding: base64

Content-Disposition: inline;

filename="tv\_newFiles.zip"

UEsDBBQACAAIAGN1UisAAAAAAAAAAAAAAAAJAAQATUVUQS1JTkYv/soAAAMAUEsHCAAAAAAC

AAAAAAAAAFBLAwQUAAgACABjdVIrAAAAAAAAAAAAAAAAFAAAAE1FVEEtSU5GL01BTklGRVNU

Lk1G803My0xLLS7RDUstKs7Mz7NSMNQz4OVyLkpNLElN0XWqBAkY6xkoaASX5in4ZiYX5RdX

Fpek5hYreOYl62nycvFyAQBQSwcIM/k1pEQAAABEAAAAUEsDBBQACAAIABZoUSsAAAAAAAAA

AAAAAAA3AAAAc3JjL2phdmEvb3JnL2FwYWNoZS9sdWNlbmUvaW5kZXgvU2VnbWVudFRlcm1W

ZWN0b3IuamF2Yd1YS2/bRhC+61dMdYglR6Wcq19piiRAksIOErc9GD6syKW0Dclld5eS3cD/

vTP7IJek5CAoUKAlIJgczs7rmxdds/QLW3OQap2wmqUbnhRNyiueiCrj92cTUdZSGfiDbVki

ZPLu+s19ymsjZNV/1xhRJMdnk8ny+HgCx8tJLipWQFowreEzX5e8Mjdclb/x1EgFeLTgRNNA

1LeK/+neLOzzR6kF6fDcXyeAl6gM5IIX2VVTnrUUg+xI0Ld3fRqJHBE/Knl/e0dUSx7ZNYt1

LAbyF0PZi5HcuTeVLrMROgmy4GJgessR5CNHuN3DYTV6Fnu/h4eM8Cx06zgeJ87VulkVIgWH

ymejRLUGI93NLDbbkX5u8pwr0CsUWPFdjzqbd8r1CtOm5lU2O/p6tJce3J4HwvQUphFnjvja

sIuLkzMQ5yEIScGrtdkg6fnz2D4LZ45HLk/mkZrpoie1b0OLobhrzThaHs3j9w5TZDgbqWqD

+wMGoymKObhzN54+i6KAUhddSvTEPe6L2mMcNcVNoyp63UHzHTC+o5L9xFmGyAk1n7jsUHKn

IapbS/4/4k1wgEEHhErW3FZ1D/k+dwgev6cj+HPYwkuY/nr14er696vZFJ6DwN90PoVTMAnx

Hk6xe/PfzS2bWkpsmeE+t7ZSZENTBqni2l9tW1+MGLlVRy7Bs2eWyyMMl3AyBPhQWu1JGDiP

he1NlzZlnKpI+GIovK+8HsW1H9un4vvYq1OcgoB5t8JAydxipjFaWI3cPsDWjpsE3uVEUxwY

/irZ43Sij8EzLzyEGk4S/2o57gkUKC3+4r2e7rHvRk2b6yeU2IMKjJOC/HilFHsIbuBB8sra

yDRuChlVkVTYcw5703pC/g+9IUueduj2Dnw9u5Dqp5z7hvk5lmLDq1RwncAvMmXUErV9vSG7

iTeVSnFdyyoDWSFgkv50PsgOxigY3XG5MkxUPENdsoTzVGb8sm//+dISW5ElNxuZJfAGFzAo

vFF9oamsSKq2pKrLLFFybVeAzj4yTKZp09mVybShRQdx6j/bZv0v4Pa2i/oh7KKtJoC33LML

drh+8nawKsQ9h9rzWs93G4HhbLESGltJU2VJ69WNdZN8FhlGQ2A0Mlg92CN2/QVmvBRhdIg6

VT9ToyKN8uF78yDg38p6qiv8IxxCLLVrqOTjATjcMtn2CfvnNBo+dPSuv5t0oBAmLoBxzwvF

4qLjdH1PlbRgtBJ3wrgnXfPUwecEmVZKCLFHzef6sFwyiVVUSePZeqUXIg0/vvhG37UuX+ez

6LMBLYgDvFzeXL++Pu0+f4BBJmie0r1G1egeDbz3rz/Ai+RFN1PxNfYk3FRsO9PJCrWqh8/2

RLvlLGK1o9FP5y8vsOW/tLen6NCoWb5vtIFCfOE+kJFL8xDKVRNM5doay+KGRKkaQ2bhYSW3

rSoZV63ZINOG6Y6RxhewuNO18tpZ6vVnpH8BnPqmLhA/3ybtZofH0c+mMHTMHSA6noj6GjNx

ciYHhl1JYVmhafi9i0r3zD4vspX8U80UKyM8dL+Ru90zUoFjpZDyS888L0Qbpky/oJysnW0S

9FwIbYLVQZ89pofCcMgPZ0h0xosncd9qKdYerkf53n4YW/3uFpX2ywBcHbguF6LLoJT4yPNc

4KTAYzsHRSYdnvidG4tw1yuSzjQq2nEMblPg2pptWZX6wMgdermixh+GfLsG2SCOJWLq7ewa

nMGWFSILnmJyFFqiQwZKsd5Yk3HgaLEq7JJQ8C1XbM3HEsl0wLeVcxBHjI0NeeUDXoosQyl9

C7us2WMkWmc2jcb2sRVZyPe+a1CyhxWNMqyvInfd0paKnYPOrWTYYhBZ9wlIKCNu4X8ldI1X

cWTYu4KTIEGSQgOJ8+PAl0vXpkJbevwbUEsHCFsYJVqKBQAAnxIAAFBLAwQUAAgACACjfFAr

AAAAAAAAAAAAAAAANAAAAHNyYy9qYXZhL29yZy9hcGFjaGUvbHVjZW5lL2luZGV4L1Rlcm1G

cmVxVmVjdG9yLmphdmGdVU1v1DAQve+vmCOtllScWyE4tFIRoqggLlUPXnuyMU3s4HG6BcR/

Z8Z2NknL8rWXtb2eN2/evPH2St+pLYIP20r1SjdYtYNGh5V1Bh9OV7brfYjwWd2ryvrq8ur8

QWMfrXenq9XJ8TG8D/7eGiRQWiMRRA8UfUADEUMH96h5B74G5QBdtAHBeD10vOasKzgGUNNJ

bbE1FZ+erPph01oN1jFOrTTCR8a7CPjlU4b8vloBf4TDu6HbYEoiOYljIDY4J1CluwIrXxM0

kP2Gz45OJ6zXIaivIxS4hJwQFbEuxrot0zZ4EPHmFrYYhWtmRQn9AHzN5QzotEWq4K3XSoSl

9DPzV+mu9iEg9d4Z8A5FX/4qyUG2+1JnZKdwv4nKOm5HHXwHZ9obfLkkeHaSDveQHcbGmwrO

2Q7QFlJLUO2doFI6cpP6tkM5szTxE2Je62HiNWv/cj82/w+6XkyiLbW9xjgEbhQbMEifAvas

HCPnCoqqvzHEhxwXfV78P/oaNkOEgTCd9XlEDFzKTF2jYvvMG5hbxFI9sAYsCorwSTjFTbdO

t4PB0ma+wrpHxnjs0b+qaMYAbDhi0OB3BMuxflKydJ/jFnM1mi0bIqSb/+IyFWHXWCl0RNzZ

mHfUo7ZsBlOA4h6lgIDqe1RcMlzWv7Cb8exC52O5trDuulCF5y8Ovgmp2Kv6mayn3AszvBko

QmvvsFCchRyNJMUCxPn5UaXUVzUfFXmo5mKkwlWHaYiqIj2J9lnh2PClRtF0UZ4uUPMZ3OPt

X8KS30j+dTYWtaxMGeDkDQ5nIw9tlLAcIOe1n5lUks/aPkq3v/CqV0F1M7Vo+VKkRHPbsO1b

7+8WWQoIRRXi0nEZa9dgyHPQWopjkWO+FEaPwVp0jx+pWUyBF7jDz05igvTEEHRzu87/IZI5

Lzmd+OTHT1BLBwjzTXdctwIAAFgHAABQSwMEFAAIAAgAWJNNKwAAAAAAAAAAAAAAADgAAABz

cmMvamF2YS9vcmcvYXBhY2hlL2x1Y2VuZS9pbmRleC9UZXJtUG9zaXRpb25WZWN0b3IuamF2

YW2QwU7DMBBE7/mKOUIOyQe04gZHhFDFBXHY2JvGorGNvYFWiH9n4yYFJHyxvDu78zyRzCvt

GSHtG4pkBm4Ok2HPjfOWj5uqausat0dhbzO2Jli+2XEa7xK/PbGRkLZtKUICYgrvzjLIWicu

eDrA+T6kkeYXqAuTVKiBGHIRZO3jY3BmAKs3Qg8ZGKIG2srow+RtoyNtFafu4IzqtdmTYcwU

D8ueMwl4wfwLiM+qgp75I48sU1JbUpqU6DQ7/gOzQvxmmI+i7wobJYb+1IvrHVt0pzJSIgPJ

ssVJhp/GjhMoRqZUHFR4WVY8VsmZpxNyXlf2KYxL3nuW2fW+6PKa98gyhAtYW+6fkJ5fsIyt

GeUrLZ8RrzfV1zdQSwcIM41IwhgBAAD5AQAAUEsDBBQACAAIAK1uTysAAAAAAAAAAAAAAAA3

AAAAc3JjL2phdmEvb3JnL2FwYWNoZS9sdWNlbmUvaW5kZXgvVGVybVZlY3RvcnNSZWFkZXIu

amF2Ye1YS2/bRhC+61eMc6lky7RzC6oIPeQBGAiSIAnSg6HDmlxKW1O7LHclWW3y3zuzLy5F

SnaCHnqoATt8fDuPb2a+XaZm+T1bclDNMmM1y1c8qzY5lzwTsuAPs5FY16oxA++1UQ3PXouG

53i1fxR5I+uN+WwaztYR+wfbskyo7ObDm4ec10YoORuNrs7P4cuH1x9+hYZX7AH0XuarRp2N

4PxqVArJKsgrpjV84c36q3WvP3FW8Ab+Ho0A6kZsmeGQeASzxVyOvSqOvyqPv6rTV0Ia0OIv

nj57K3hV3MhSaSjj5Ywi7AU+/syXay6NT6NmDd5MIbILxRTQsZBL0A45HTY/AYNU7TQklCIr

gD9mJXTWImF+GBWAKGFcIKbibx6ENnrsncEFPMuQwmeTiTdmDW4f0EiRqZrLt7imj54l2OIU

tuhiy1PYsoutT2FrwkYwFQjRYyzWBOPLKi6XZjWewBW8cCa/j+gX/7g2850ncV0BWyUKbDyl

+fgEy1dXsFthvZT8xcCOg2lYjWgOPAKZLIAZqDjTBtbsnoPeNIhcMRNMsAqd20bT9jlZypl0

7gG7w10Vv8WyUTHO5iA3VTWhymQ+0lmCKFJEMYQoU0Q5hKhTRJ0iEt7CNIxDuzTcbBoZJ+Q7

DXlE+95vhwKW3FCRkMX8/WYdW9/26xTulELuJKE+Ki2IUz0ZwUBN8GGsyqsVz+8pCanAoCvY

ugG0bKIOiYrdVRxKdE+TEuaMKoXV6PA8Dwz4tOjGt1nqyJqxMQNeCLllFXYQ2l8xTVFYXSzS

YKyJONe4tjOk7fCO7VVSF0K6oODbNzgrhRPdltKhWP0TIjsQjQvlZn3Hm2mXXVc0lN/h0Xi8

gnRtg37vzf+Xi0jjozm/97TAObx458muFPZh7QPG6hAUx7R4h8/Hk7i+cOsDMFQqkPBKbfBy

boG0+uuNNHE1pvllxRs35kpWe2BQ8p1bqaHG/QHj2lBeGfyOkNrYhBFF9deoE8FOw1B5iAou

Mc0/N8KOkRUUfLaHO9r4ccPhhbUkObWjAgrJFmKzXKWKpEqrZD4QJvc7tic81pD+oXckIbXC

RHmjs5i06ylM+DohAjmgPeHyuXtGKdiWEfPrGQh4mVCF9xcX6dbjDV70KQwIKnNwO09bbxI9

i6D5Le1hYKeYr1lR/uCmVkibXtx8d3aEQxb2La4NdWnn0rmiLI+phVrjoYK3PeU9EZG0EVtc

0nDXR9iau1AcU5GFuDBlqtOrPih60Y7vuDOqwcigJpAVOrB9cnaoTTqD6UUuTmTbuqgmTxOT

20VPTv5XjxPq8V7FEcW8XI6W2V4Rkv2jtYs9NpTWkTnG4oRmoSOlRKGip63BhV//L893dHor

FuTXLjgc6J8cLioT5oWoj17KfGb2RSe1H87s5EQGUOLZpRdWdRIcHF0dZ1dPUztHh9dNYPhm

GRq+QwfpXq5vF1PX1mnMi5+a0P4pouFoy3PfjyzG4A/ybTmSarw8RPUqQk4syUc0kN5OD0oy

wKb7eEiqop/M8aHnlOAeuz9MbSsLu7CzcyiYwfNCo9aga54L9FbEJvMqVDoVaj1PukpAAWsr

Q+WgDN1EndVwGWhhkGO0hqEJvq7N3r5PDr9+2/a2O1I01APjliaSqvRvGq2rvJMJZ/piOGwC

U0SusxM9CwsXsw7uLZ6qjgLjKYxIL0WDX3t4mNhw7QgObq4XAyQGhHMwCDnUnudOe0IAvT6P

Dm2zt3eXzxcDdKSrXBDiWJxejvptbbbHZjepWwhk2rqKBQwdcdZp9dARZttyLMrYvn47r7Sy

Z16uDfb2JSz9OXXF150SfmzUw+1iuIa3oYpu40akH4UTFbP2roOe0N04WTjtVPUnqpgGcTEc

RScS8ZRIxKJbSrPNggHfKnQ5S/WNyA/ylvwnmN8urLPOacd9Aua0Jz6yDRAuFX6ymifbLTVE

3j+qkODaVtB4bsdPbsm3nD6165rLkFU9eLAiZ37C6h6Z/dLkg/t6soskM9W3d7h3h13i+z9Q

SwcIuCMrw+MFAACKFQAAUEsDBBQACAAIAIRwTysAAAAAAAAAAAAAAAA3AAAAc3JjL2phdmEv

b3JnL2FwYWNoZS9sdWNlbmUvaW5kZXgvVGVybVZlY3RvcnNXcml0ZXIuamF2YcVaS3PbOBK+

+1cgvoRKZDp7djxTmWSy66qtJLXxZg5TOcAkaGFNkQwBytam/N+3u/EmKUtxklkdbIoA+oXu

rxsNdby44deCtf11zjterEReD4VoRC6bUtydHcl11/Z6Zlzpthf5G9mLAp62e2e+H3Q36I+6

F3ztJ/+Hb3gu2/zi/e93hei0bJt0bNCyzj8Ri7Ojo6NKNrxmRc2VYpeiX5sR9UcvtejZ16Mj

Bp+YE9ObO3bOmqGul/BcRs9V9NzZ5zMiYKiySoq6VHMjwGwdBmjk9JTdCNHJ5prplVSsM4Y9

2Uglr2rBbmVdM17X7S1bC71qgbBu2ZVgsqllI8pficrlp7fIlBVD34tGmy+xAHULDOzom7b4

0MoGNT9nJ39zgjx7xl6D6lqQmGxjTMRujY0qFH8lmOpEIUHDkilxvQZqIEj0snT7mhNR9oyx

V6wRtzNWV6t2AClBlYLYlsRDgAewsi0GpJ2zS2DZ8Z6D7qL3JF8WbSl+WfM7UlS9PKXvIEkp

C6Ck2ArNxZst2ErDxtsd4b0AHkNTwkxPi6weGL5rNZqbtRWqq4Rbu+Zb1osvA+hH9nHmMZtB

rlouPVHVkrGaYX0FmgKtAmjSZCN624mGZPeiK1YLcE60wJcBRNatkcyQcNY8pf/dcFXLYmrS

zEdV2IclA4dG77LbtQTdNfOmWxBBverbW8WiaKLXXw1X9NH3IDCYAmSkXQqsSXVwWDvVhE3w

ArO1b2Fh5vzlOTvOYdrx4syvKQ9bU8ZrqsPWVPGa7rA1Ha6xi0IwgxMblbNgPU/aBXaYZMfu

jwwlg0B25zatLBn6wBvrd9lB+8AAwVolwiIvJYYve/P7b//+u/MRs28ft0qLdd4OOu/AC3Td

ZMcp32CelEZgOQMasBX5tdBoOvtyv7oj0Q/TV1Ysk8qtQifMFoto3NuE9iMxyC6j7DbMSMLY

Mjutgx9CSFiXjRYY18mLWvB+PLYLit34/W5rXrUtUGzY2C6RWXqhh76ZYfIkcHG0Efc/ag6J

s+vbAjAIwYIb4RF/AYMK4HaF0IlpCHD6agsTptjm0Y+XJQLEknGAWm6lx1mIamG78JuDbIJL

AdPbypOJ5CE0BkmsVBeVk5DdcsiZvdjIdlD1loIKIA6ykvJ0iCMIMuh2zbVEQbYpoM4Hp/Ep

hEti9Y4U/gbHfTL1XFpHKHFR1+Ka12B5LTyR7Pg1bxrIQK2BW1JwBY9N61MUpgoczo9jb0+D

AAA7kiMpCp6YqgC9JcwyMcEU39h8RztHoDaaRb4+wyfyd9j9hOVkHtENYREG7x8LZ0Ygv00Q

S8cA5dGuIbAfDHW+esKyxRRWWewBPn5c+LyVjVQrcLHIYy2tJI5CxZOGkg0N77BeJZgK2Ram

DxDPIhQRNsD2OXHsFN+Ct7RkHmwfgahWgIPhdJcj7nbDw5zwAReMh3aW0GbTR1v/L4O0uh8E

Gs+hEoKmoWMxKZ/dpQDlkcl34ngSwBMnfFUaQ5nK0ZrvqYqL+ZwZEutBYZEL5U+5RW9EAYPz

Vewd0LcZwzsKqkTuA6pcUrUTvBmMLZKaWjZYWxnk7kuTJVAOCCC1ZG1jXJl1MDI08ssg3GjO

PvTtneUdyEHpzjqbwuCvURCl+NAqSdJhacqyrr9bgJJQpxtxowo81PtyLZTJJmQa3nXgBQqP

MSHFLP1XfzLYE2s2JCldWGVMqV2BNOaps8KqPz8vflyq4HbflckUcZpwmSQOPcstCfFHsvKu

HvPxjKxBLnDTwFiZN4oxiDdGCqcAHRssB4yBY8t6QodbeF82oYPRb0NV4Wn0yiJ+/DbGhtNT

NJ3nECfSCLnQN+BQRELK8xdnTL70S/JaNNd6Be+eP58uc5sj2S/sxQLkydEzmzJ7unyaiOGe

woygtvw8mXoffY+f5/DaObGzr02k7hskUbN9LsHiI70MZjEjYM1vzrjWPOz5OREOKy8/XRKu

4R+XlUnMiDoO5jpGhHMn9mhSLKp/Hs2xOqZimJyBiQWfFtMqGhE4dEKiBgEC0TfgiD3TZx/N

oTSctf/87Kj+NVWoi/fQCzoUYb4TYFKGHmcmGDMONbtsV6Dhx9vX44ldlMbO/UG7O9lc5kxt

cqzNrexq0FjCkyKhR+TzJ83Bzs9sssWmG18HJnQcW2GFZHpJNmhtJykcwqBw8hOsS7lukykk

1/JOQKZbQSG6AQTk7FpuvLlJfiFBux5EDGeythtqatbdwtisBKhJOmi42+HckDo0DKZRYG3/

fwmBvzoCdgTAPmcegdN8Uh2t/E47h2OYrTXdiSkSF8J1Gq25kv8VAOU7gjVJ/XaFKyUhZJcs

eoeF3+QlFpRzsY2f9NQ8jvjXOErNYEV3AmpUxIczVraYMVCki+63c02ruJHnhTPE6/ECSOZr

fgPiNExUFd5zuF5KLCG7FaZbA4DTi7Wg4heRoBcnJOGYpjkp9BjrXm7RFO2AJqei3l5LmJNt

sjzWFe8wJqclFwvYFA71EhkDG8W5td0ZKF1wXaxYFpMU4A203NB2yy0ncRbtY2BUzjAqfwaj

aoZR9TMYdTOMuh/PiMYCI4KrmKZZDVFd1xcNgFdxc9nzQmSzOTNBHQqT+MC+H0zAMb33AgLi

lYs/miEOIjXtzqxsVEbCPsSd6mqmUx0QiQIZMQlRKBrAdSTzJwCfDAdtQaksXsW1ZzJ3rqQ9

IfJJu/7UmMSAvcTLKYhSX4+GVAKi1Vzpy7S0fZGAalIDkSJzYJqW0Zn5urBKiVogEr3SmRy3

ShLtpnX2yUS+/QSwsB736me0nDCLfW2HKQGo7lyvQMVzLuhCD1yJ2xs8tbQvsL1Q3/Kt8vZn

Jdc8Z/8wldEyJgOxksxy919QJBUtdhZacmjwKoBiW+IVK1HcMOS1TF28aHtqkNecWtwKCLAb

2ZSmc471APVJSD1nA5Utck+FrnXRclHb5Oe7R7pvKAJa/YNG1iNZ0yPXTs/4JxDJHJGTsUZz

rpIqbJc+VL+7Th1eKNt1XAO6SABPc9MbToZUXV8JKL0s0oTbbONG0Vze93yLPiTWnd4ucddW

2PgawxS2nlxnrhah7I4FMlgXaN+S+0ykiS83TItQlKMbYou9tDmjLZk0Sh4sXcz2aoOl3Y5b

P/w4pDoIpdhLtrctYjyki7Aj7nM4L5nHnJEg8cK5OtD2WTvnQePCOaQwuuZ7VB997wngrctu

SuNvPuhQCO6JvwEBzuhS7txxPJtJwqkEf4Fj0SVKUnfGkhRqk5vBhykS0I0I7kydAY+TX0CY

xnwkTzlNsLZ57zPslKYpAQzVaZZ8G934PBYFzTZUtv2f2RcLJ9tDiTLWyNz7WPVPxtLNIVoq

fLz+4bRnRJ2kPJ8aiO735oYfYBVyvGpcop1MZNxpm6DDmMpsDZqGsNJ49et+Amb1+Zp40NjY

ZMGkmnwxgjtT3eF7syHB20sJUdwU2hYlvq6Lmprhetksw5ody1znNOdOoPTWaYdClLxTffTO

mtF1zadvo8TARhdNR/f/A1BLBwgTn4NwOAoAAPUnAABQSwECFAAUAAgACABjdVIrAAAAAAIA

AAAAAAAACQAEAAAAAAAAAAAAAAAAAAAATUVUQS1JTkYv/soAAFBLAQIUABQACAAIAGN1Uisz

+TWkRAAAAEQAAAAUAAAAAAAAAAAAAAAAAD0AAABNRVRBLUlORi9NQU5JRkVTVC5NRlBLAQIU

ABQACAAIABZoUStbGCVaigUAAJ8SAAA3AAAAAAAAAAAAAAAAAMMAAABzcmMvamF2YS9vcmcv

YXBhY2hlL2x1Y2VuZS9pbmRleC9TZWdtZW50VGVybVZlY3Rvci5qYXZhUEsBAhQAFAAIAAgA

o3xQK/NNd1y3AgAAWAcAADQAAAAAAAAAAAAAAAAAsgYAAHNyYy9qYXZhL29yZy9hcGFjaGUv

bHVjZW5lL2luZGV4L1Rlcm1GcmVxVmVjdG9yLmphdmFQSwECFAAUAAgACABYk00rM41IwhgB

AAD5AQAAOAAAAAAAAAAAAAAAAADLCQAAc3JjL2phdmEvb3JnL2FwYWNoZS9sdWNlbmUvaW5k

ZXgvVGVybVBvc2l0aW9uVmVjdG9yLmphdmFQSwECFAAUAAgACACtbk8ruCMrw+MFAACKFQAA

NwAAAAAAAAAAAAAAAABJCwAAc3JjL2phdmEvb3JnL2FwYWNoZS9sdWNlbmUvaW5kZXgvVGVy

bVZlY3RvcnNSZWFkZXIuamF2YVBLAQIUABQACAAIAIRwTysTn4NwOAoAAPUnAAA3AAAAAAAA

AAAAAAAAAJERAABzcmMvamF2YS9vcmcvYXBhY2hlL2x1Y2VuZS9pbmRleC9UZXJtVmVjdG9y

c1dyaXRlci5qYXZhUEsFBgAAAAAHAAcAdAIAAC4cAAAAAA==

--------------010909000801060505060408

Content-Type: text/plain;

name="tv\_diffs-1.2-rc1.txt"

Content-Transfer-Encoding: 7bit

Content-Disposition: inline;

filename="tv\_diffs-1.2-rc1.txt"

? tv\_notes.txt

? src/java/org/apache/lucene/index/SegmentTermVector.java

? src/java/org/apache/lucene/index/TermFreqVector.java

? src/java/org/apache/lucene/index/TermPositionVector.java

? src/java/org/apache/lucene/index/TermVectorsReader.java

? src/java/org/apache/lucene/index/TermVectorsWriter.java

Index: src/java/org/apache/lucene/analysis/PorterStemmer.java

===================================================================

RCS file: /home/cvspublic/jakarta-lucene/src/java/org/apache/lucene/analysis/PorterStemmer.java,v

retrieving revision 1.1.1.1

diff -w -r1.1.1.1 PorterStemmer.java

95c95

< class PorterStemmer

---

> public class PorterStemmer

Index: src/java/org/apache/lucene/document/Field.java

===================================================================

RCS file: /home/cvspublic/jakarta-lucene/src/java/org/apache/lucene/document/Field.java,v

retrieving revision 1.1.1.1

diff -w -r1.1.1.1 Field.java

73a74

> private boolean storeTermVector = false;

89c90

< fields, like "title" or "subject". \*/

---

> fields, like "title" or "subject". Term vector will not be stored for this field. \*/

91c92

< return new Field(name, value, true, true, true);

---

> return Text(name, value, false);

95c96,103

< but that is not stored in the index. \*/

---

> and is stored in the index, for return with hits. Useful for short text

> fields, like "title" or "subject". \*/

> public static final Field Text(String name, String value, boolean storeTermVector) {

> return new Field(name, value, true, true, true, storeTermVector);

> }

>

> /\*\* Constructs a String-valued Field that is tokenized and indexed,

> but that is not stored in the index. Term vector will not be stored for this field. \*/

97c105,111

< return new Field(name, value, false, true, true);

---

> return UnStored(name, value, false);

> }

>

> /\*\* Constructs a String-valued Field that is tokenized and indexed,

> but that is not stored in the index. \*/

> public static final Field UnStored(String name, String value, boolean storeTermVector) {

> return new Field(name, value, false, true, true, storeTermVector);

102c116

< "body". \*/

---

> "body". Term vector will not be stored for this field. \*/

104c118,127

< return new Field(name, value);

---

> return Text(name, value, false);

> }

>

> /\*\* Constructs a Reader-valued Field that is tokenized and indexed, but is

> not stored in the index verbatim. Useful for longer text fields, like

> "body". \*/

> public static final Field Text(String name, Reader value, boolean storeTermVector) {

> Field f = new Field(name, value);

> f.storeTermVector = storeTermVector;

> return f;

117a141,144

>

> /\*\* Create a field by specifying all parameters except for <code>storeTermVector</code>,

> \* which is set to <code>false</code>.

> \*/

119a147,153

> this(name, string, store, index, token, false);

> }

>

> /\*\* Create a field by specifying all parameters.

> \*/

> public Field(String name, String string,

> boolean store, boolean index, boolean token, boolean storeTermVector) {

123a158,159

> if (!index && storeTermVector)

> throw new IllegalArgumentException("cannot store a term vector for fields that are not indexed.");

129a166

> this.storeTermVector = storeTermVector;

130a168

>

153a192,199

>

> /\*\* True iff the term or terms used to index this field are stored as a term vector,

> \* avaliable from IndexReader.getTermFreqVector and IndexReader.getTermPositionVector.

> \* These methods do not provide access to the original content of the field, only to

> \* terms used to index it. If the original content must be preserved, use the <code>stored</code>

> \* attribute instead.

> \*/

> public final boolean isTermVectorStored() { return storeTermVector; }

Index: src/java/org/apache/lucene/index/DocumentWriter.java

===================================================================

RCS file: /home/cvspublic/jakarta-lucene/src/java/org/apache/lucene/index/DocumentWriter.java,v

retrieving revision 1.1.1.1

diff -w -r1.1.1.1 DocumentWriter.java

62a63,65

> // for TermVector support

> import java.util.Vector;

>

265a269

> // Inverse index files

268a273,275

> // TermVector index files

> TermVectorsWriter tvw = null;

>

269a277

> // Open files for inverse index storage

271a280

>

274a284,289

> // Term Vector support

> tvw = new TermVectorsWriter(directory, segment, fieldInfos.size());

> tvw.openDocument();

> String currentField = null;

>

>

297a313,324

>

>

> // check to see if we switched to a new field

> String termField = posting.term.field();

> if (currentField != termField) {

> // changing field - see if there is something to save

> currentField = termField;

> FieldInfo fi = fieldInfos.fieldInfo(currentField);

> if (fi.storeTermVector)

> tvw.openField(fi.number);

> else

> tvw.closeField();

298a326,334

>

>

> if (tvw.isFieldOpen()) {

> tvw.addTerm(tis.position(), f, positions);

> }

> }

>

> tvw.closeDocument();

>

301,303c337,344

< if (freq != null) freq.close();

< if (prox != null) prox.close();

< if (tis != null) tis.close();

---

> // make an effort to close all streams we can but remember and re-throw

> // the first exception encountered in this process

> IOException keep = null;

> if (freq != null) try { freq.close(); } catch (IOException e) { if (keep == null) keep = e; }

> if (prox != null) try { prox.close(); } catch (IOException e) { if (keep == null) keep = e; }

> if (tis != null) try { tis.close(); } catch (IOException e) { if (keep == null) keep = e; }

> if (tvw != null) try { tvw.close(); } catch (IOException e) { if (keep == null) keep = e; }

> if (keep != null) throw (IOException) keep.fillInStackTrace();

306a348

>

Index: src/java/org/apache/lucene/index/FieldInfo.java

===================================================================

RCS file: /home/cvspublic/jakarta-lucene/src/java/org/apache/lucene/index/FieldInfo.java,v

retrieving revision 1.1.1.1

diff -w -r1.1.1.1 FieldInfo.java

62c62,65

< FieldInfo(String na, boolean tk, int nu) {

---

> // true if term vector for this field should be stored

> boolean storeTermVector;

>

> FieldInfo(String na, boolean tk, int nu, boolean storeTermVector) {

65a69

> this.storeTermVector = storeTermVector;

Index: src/java/org/apache/lucene/index/FieldInfos.java

===================================================================

RCS file: /home/cvspublic/jakarta-lucene/src/java/org/apache/lucene/index/FieldInfos.java,v

retrieving revision 1.1.1.1

diff -w -r1.1.1.1 FieldInfos.java

68a69,74

> /\*\* Access to the Field Info file that describes document fields and whether or

> \* not they are indexed. Each segment has a separate Field Info file. Objects

> \* of this class is thread-safe for multiple readers, but only one thread can

> \* be adding documents at a time, with no other reader or writer threads

> \* accessing this object.

> \*/

74c80

< add("", false);

---

> add("", false, false);

91c97

< add(field.name(), field.isIndexed());

---

> add(field.name(), field.isIndexed(), field.isTermVectorStored());

95,96c101,106

< /\*\* Merges in information from another FieldInfos. \*/

< final void add(FieldInfos other) {

---

> /\*\* Merges in information from another FieldInfos.

> \* Returns an array mapping each field number in the <code>other</code>

> \* fieldInfos object to the field numbers in this one.

> \*/

> final int[] add(FieldInfos other) {

> int res[] = new int[other.size()];

99c109

< add(fi.name, fi.isIndexed);

---

> res[i] = add(fi.name, fi.isIndexed, fi.storeTermVector);

100a111

> return res;

103c114,119

< private final void add(String name, boolean isIndexed) {

---

> /\*\* If the field is not yet known, adds it. If it is known, checks

> \* to make sure that the isIndexed flag is the same as was given

> \* previously for this field. If not - throws IllegalStateException.

> \* Returns the number of the previously existing or a newly added field.

> \*/

> private final int add(String name, boolean isIndexed, boolean storeTermVector) {

106c122

< addInternal(name, isIndexed);

---

> return addInternal(name, isIndexed, storeTermVector);

110c126,133

< " be an indexed field.");

---

> " be an indexed field (already defined otherwise).");

> else if (fi.storeTermVector != storeTermVector)

> throw new IllegalStateException("field " + name +

> (fi.storeTermVector ? " must" : " cannot") +

> " store term vector (already defined otherwise).");

>

> else

> return fi.number;

113,114c136,138

< private final void addInternal(String name, boolean isIndexed) {

< FieldInfo fi = new FieldInfo(name, isIndexed, byNumber.size());

---

> /\*\* Returns the number of the newly created field. \*/

> private final int addInternal(String name, boolean isIndexed, boolean storeTermVector) {

> FieldInfo fi = new FieldInfo(name, isIndexed, byNumber.size(), storeTermVector);

116a141

> return fi.number;

155a181,183

> byte bits = 0x0;

> if (fi.isIndexed) bits |= 0x1;

> if (fi.storeTermVector) bits |= 0x2;

157c185

< output.writeByte((byte)(fi.isIndexed ? 1 : 0));

---

> output.writeByte(bits);

163,165c191,197

< for (int i = 0; i < size; i++)

< addInternal(input.readString().intern(),

< input.readByte() != 0);

---

> for (int i = 0; i < size; i++) {

> String name = input.readString().intern();

> byte bits = input.readByte();

> boolean isIndexed = (bits & 0x1) != 0;

> boolean storeTermVector = (bits & 0x2) != 0;

> addInternal(name, isIndexed, storeTermVector);

> }

Index: src/java/org/apache/lucene/index/FieldsReader.java

===================================================================

RCS file: /home/cvspublic/jakarta-lucene/src/java/org/apache/lucene/index/FieldsReader.java,v

retrieving revision 1.1.1.1

diff -w -r1.1.1.1 FieldsReader.java

65a66,68

> /\*\* Class responsible for access to stored document fields.

> \* Uses the seg.fdt and seg.fdx files.

> \*/

Index: src/java/org/apache/lucene/index/IndexReader.java

===================================================================

RCS file: /home/cvspublic/jakarta-lucene/src/java/org/apache/lucene/index/IndexReader.java,v

retrieving revision 1.2

diff -w -r1.2 IndexReader.java

219a220,261

>

>

> // TermVector support

>

> /\*\* Return a term corresponding to the specified term number.

> \* If no term with this number exists, return null.

> \*/

> abstract public Term getTerm(int termNumber) throws IOException;

>

>

> /\*\* Return an array of term frequency vectors for the specified document.

> \* The array contains a vector for each vectorized field in the document.

> \* Each vector vector contains term numbers and frequencies for all terms

> \* in a given vectorized field.

> \* If no such fields existed, the method returns null.

> \*/

> abstract public TermFreqVector[] getTermFreqVectors(int docNumber)

> throws IOException;

>

> /\*\* Return a term frequency vector for the specified document and field. The vector

> \* returned contains term numbers and frequencies for all terms

> \* in the specified field of this document, if the field had storeTermVector flag set.

> \* If the flag was not set, the method returns null.

> \*/

> abstract public TermFreqVector getTermFreqVector(int docNumber, String field)

> throws IOException;

>

> /\*\* Return an array of term position vectors for the specified document.

> \* The array contains a vector for each vectorized field in the document.

> \* Each vector returned is like a term frequency vector but also contains positions

> \* in which the terms were found in their fields.

> \* If no such fields existed, the method returns null.

> \*/

> abstract public TermPositionVector[] getTermPositionVectors(int docNumber)

> throws IOException;

>

> /\*\* Return a term position vector for the specified document. The vector

> \* returned is like a term frequency vector but also contains positions

> \* in which the terms were found in their fields.

> \*/

> abstract public TermPositionVector getTermPositionVector(int docNumber, String field)

> throws IOException;

Index: src/java/org/apache/lucene/index/SegmentMergeInfo.java

===================================================================

RCS file: /home/cvspublic/jakarta-lucene/src/java/org/apache/lucene/index/SegmentMergeInfo.java,v

retrieving revision 1.1.1.1

diff -w -r1.1.1.1 SegmentMergeInfo.java

59a60,63

> /\*\* Data container to work with SegmentMergeQueue. Represents a single segment

> \* to be merged. Maintains the segment reader, TermEnum, and TermPositions

> \* for this segment.

> \*/

60a65

> /\*\* The current term of this segment, or null if none. \*/

61a67,68

>

> /\*\* Index of the 0th document from this segment in the merged document numbering. \*/

62a70,71

>

> /\*\* This segment's term enum. Do not use directly. \*/

63a73,74

>

> /\*\* This segment's reader. Do not use directly. \*/

64a76,77

>

> /\*\* Postings for the current term. \*/

65a79,85

>

>

> /\*\* Maps around deleted docs. Contains a slot for each document in the

> \* reader. Slots corresponding to deleted docs have the value of -1. The

> \* rest have their new document numbers that start at 0. This value

> \* added to <code>base</code> is the document number in the merged numbering.

> \*/

67a88,100

>

> /\*\* Maps term numbers from this segment to term numbers in the merged segment.

> \* This map populated during term merge and used during term vector merge.

> \* The variable is in this class only to help associate the map with the segment

> \* it belongs to.

> \*/

> int[] termMap = null;

>

>

> /\*\* Create a new merge info. Base <code>b</code> is a starting

> \* number for documents from this segment in the merged document

> \* numbering.

> \*/

88a122,125

>

> // Allocate term map and set all entries to -1 (0 is a valid term number)

> termMap = new int[te.size];

> for(int i=0; i<termMap.length; i++) termMap[i] = -1;

90a128,133

>

> /\*\* Shift to the next term on this segment's TermEnum. The new

> \* term becomes the current term for this segment, effecting the

> \* ordering of the SegmentMergeQueue. If no more terms remain

> \* in this segment, returns false and resets the current term to null.

> \*/

Index: src/java/org/apache/lucene/index/SegmentMergeQueue.java

===================================================================

RCS file: /home/cvspublic/jakarta-lucene/src/java/org/apache/lucene/index/SegmentMergeQueue.java,v

retrieving revision 1.1.1.1

diff -w -r1.1.1.1 SegmentMergeQueue.java

59a60,63

> /\*\* Priority queue of SegmentMergeInfo objects. The queue sorts the

> \* info objects by their current term, and if the terms are equal,

> \* by their base offset.

> \*/

Index: src/java/org/apache/lucene/index/SegmentMerger.java

===================================================================

RCS file: /home/cvspublic/jakarta-lucene/src/java/org/apache/lucene/index/SegmentMerger.java,v

retrieving revision 1.1.1.1

diff -w -r1.1.1.1 SegmentMerger.java

73a74,77

> /\*\* Create a segment merger that will merge a number of segments (specified

> \* as SegmentReaders added to this object with calls to <code>add</code>) into a

> \* single segment with the specified <code>name</code>.

> \*/

78a83,85

> /\*\* Add segment reader to be merged.

> \*

> \*/

82a90,92

> /\*\* Return one of the segment readers being merged.

> \*

> \*/

86a97,99

> /\*\* Start the merge. All segment readers to be merged must have been added

> \* prior to this call.

> \*/

91a105

> mergeVectors();

100a115,131

> /\*\* A list of maps. Has a map for each source reader. A map contains

> \* a slot for each field from the source reader, indexed by the field number.

> \* The value is the number of the corresponding field in the merged segment.

> \*/

> private int fieldMaps[][];

>

>

> /\*\* A list of maps. Has a map for each source reader. A map contains a slot for

> \* each term from the source reader, indexed by the term number.

> \* The value is the number of the corresponding term in the merged segment.

> \*/

> private int termMaps[][];

>

>

> /\*\* Merge the field information from the segment readers.

> \* Called from <code>merge</code>.

> \*/

102a134

> fieldMaps = new int[readers.size()][]; // allocate fieldmap list

105c137

< fieldInfos.add(reader.fieldInfos);

---

> fieldMaps[i] = fieldInfos.add(reader.fieldInfos);

129a162,164

> /\*\* Merge the term index, frequency and proximity information

> \* from specified segment readers. Called from <code>merge</code>.

> \*/

137c172

< mergeTermInfos();

---

> termMaps = mergeTermInfos();

147c182,188

< private final void mergeTermInfos() throws IOException {

---

> /\*\* Merge the term index information. Called from <code>mergeTerms</code>.

> \* Returns a list of maps mapping terms in the source segments to the terms

> \* in the merged segment.

> \*/

> private final int[][] mergeTermInfos() throws IOException {

> // Create and populate a priority queue of segments to be merged.

> // Segments are sorted by their top term and the base doc number in the merged segment.

148a190,194

>

> // Also keep refs to the segments in an array that is in the same order as

> // the source segments. This is used to populate termMaps after the merge is done.

> SegmentMergeInfo smisByReader[] = new SegmentMergeInfo[readers.size()];

>

155c201,202

< if (smi.next())

---

>

> if (smi.next()) {

157,158c204,206

< else

< smi.close();

---

> smisByReader[i] = smi;

> } else

> smi.close(); // if no terms - don't bother

168a217,218

> // pop off the queue and put into match[] all segments

> // that have the same term at the top

173a224,225

> // perform the merge for all segments that are positioned on

> // the same term

175a228,229

> // advance the matched segments to the next term and, if one exists, put

> // the segment back onto the queue (priority queue takes care of sorting them)

183a238,247

>

> // Build the termMaps array. Those that remain null did not have any terms

> // and were never placed onto the merge queue.

> int res[][] = new int[readers.size()][];

> for (int i=0; i<res.length; i++) {

> if (smisByReader[i] != null)

> res[i] = smisByReader[i].termMap;

> }

>

> return res;

187a252,259

>

> /\*\* Merge one term found in one or more segments. The array <code>smis</code>

> \* contains segments that are positioned at the same term. <code>N</code>

> \* is the number of cells in the array actually occupied.

> \*

> \* @param smis array of segments

> \* @param n number of cells in the array actually occupied

> \*/

193c265

< int df = appendPostings(smis, n); // append posting data

---

> int df = appendPostings(smis, n, termInfosWriter.size()); // append posting data

202c274,284

< private final int appendPostings(SegmentMergeInfo[] smis, int n)

---

>

> /\*\* Process postings from multiple segments all positioned on the

> \* same term. Writes out merged entries into freqOutput and

> \* the proxOutput streams.

> \*

> \* @param smis array of segments

> \* @param n number of cells in the array actually occupied

> \* @param mergedTermNum is the term number in the merged segment for the term being merged

> \* @return number of documents across all segments where this term was found

> \*/

> private final int appendPostings(SegmentMergeInfo[] smis, int n, int mergedTermNum)

211c293,298

< smi.termEnum.termInfo(termInfo);

---

> smi.termEnum.termInfo(termInfo); // copy out data into provided TermInfo (avoid mem. alloc.)

>

> // setup mapping for this term

> int sourceTermNum = smi.termEnum.position;

> smi.termMap[sourceTermNum] = mergedTermNum;

>

246a334,336

> /\*\* Merge field normalization factors for the specified segment readers.

> \* Called from <code>merge</code>.

> \*/

272a363,429

> }

> }

>

>

> /\*\* Merge term vector information.

> \* Called from <code>merge</code>.

> \*/

> private final void mergeVectors() throws IOException {

> // files to merge: tvx, tvt, tvf

> // Using TermVectorsWriter - this must work in the following order:

> // Create TermVectorsWriter

> // for all documents in the merged sequence

> // openDoc

> // for all fields in the document {

> // translate field number

> // openField

> // for all unique terms in the field {

> // convert term number

> // use new proxPointer

> // addTerm

> // }

> // closeField

> // }

> // closeDoc

> // }

> // close Writer

> TermVectorsWriter tvw = null;

>

> try {

> tvw = new TermVectorsWriter(directory, segment, fieldInfos.size());

> for (int r = 0; r < readers.size(); r++) {

> SegmentReader reader = (SegmentReader)readers.elementAt(r);

> TermVectorsReader sourceTVR = reader.tvs;

> BitVector deletedDocs = reader.deletedDocs;

> int maxDoc = reader.maxDoc();

> for (int d = 0; d < maxDoc; d++) {

> // skip deleted docs

> if (deletedDocs != null && deletedDocs.get(d)) continue;

> tvw.openDocument();

>

> // get all term vectors

> SegmentTermVector sourceTVs[] = sourceTVR.get(d, true);

>

> if (sourceTVs != null) {

> for (int f = 0; f<sourceTVs.length; f++) {

> // translate field numbers

> SegmentTermVector tv = sourceTVs[f];

> tv.fieldNum = fieldMaps[r][tv.fieldNum];

>

> // translate term numbers

> for (int t = 0; t < tv.termNums.length; t++) {

> tv.termNums[t] = termMaps[r][tv.termNums[t]];

>

> // sanity check

> if (tv.termNums[t] == -1)

> throw new IllegalStateException("Term from non-deleted doc's term vector maps to -1 during merge.");

> }

> }

>

> tvw.addVectors(sourceTVs);

> }

>

> tvw.closeDocument();

> }

> }

> } finally {

> if (tvw != null) tvw.close();

Index: src/java/org/apache/lucene/index/SegmentReader.java

===================================================================

RCS file: /home/cvspublic/jakarta-lucene/src/java/org/apache/lucene/index/SegmentReader.java,v

retrieving revision 1.2

diff -w -r1.2 SegmentReader.java

76a77

> TermVectorsReader tvs;

115a117,118

>

> tvs = new TermVectorsReader(this, directory, segment, fieldInfos);

140a144

> if (tvs != null) tvs.close();

166a171,176

> // TermVector support files

> files.addElement(segment + ".tvx"); // term vector index

> files.addElement(segment + ".tvd"); // term vector document data

> files.addElement(segment + ".tvf"); // term vector field data

> files.addElement(segment + ".tvp"); // term vector positions data

>

243a254

> synchronized(norm) {

250a262

> }

289a302,369

>

>

> // TermVector support

>

> /\*\* Return a term corresponding to the specified term number.

> \* If no term with this number exists, return null.

> \*/

> public Term getTerm(int termNumber) throws IOException {

> return tis.get(termNumber);

> }

>

> /\*\* Return a term frequency vector for the specified document and field. The vector

> \* returned contains term numbers and frequencies for all terms

> \* in the specified field of this document, if the field had storeTermVector flag set.

> \* If the flag was not set, the method returns null.

> \*/

> public TermFreqVector getTermFreqVector(int docNumber, String field)

> throws IOException

> {

> // Check if this field is invalid or has no stored term vector

> FieldInfo fi = fieldInfos.fieldInfo(field);

> if (fi == null || !fi.storeTermVector) return null;

>

> return tvs.get(docNumber, fi.number, false);

> }

>

>

> /\*\* Return an array of term frequency vectors for the specified document.

> \* The array contains a vector for each vectorized field in the document.

> \* Each vector vector contains term numbers and frequencies for all terms

> \* in a given vectorized field.

> \* If no such fields existed, the method returns null.

> \*/

> public TermFreqVector[] getTermFreqVectors(int docNumber)

> throws IOException

> {

> return tvs.get(docNumber, false);

> }

>

>

>

> /\*\* Return a term position vector for the specified document. The vector

> \* returned is like a term frequency vector but also contains positions

> \* in which the terms were found in their fields.

> \*/

> public TermPositionVector getTermPositionVector(int docNumber, String field)

> throws IOException

> {

> // Check if this field is invalid or has no stored term vector

> FieldInfo fi = fieldInfos.fieldInfo(field);

> if (fi == null || !fi.storeTermVector) return null;

>

> return tvs.get(docNumber, fi.number, true);

> }

>

>

> /\*\* Return an array of term position vectors for the specified document.

> \* The array contains a vector for each vectorized field in the document.

> \* Each vector returned is like a term frequency vector but also contains positions

> \* in which the terms were found in their fields.

> \* If no such fields existed, the method returns null.

> \*/

> public TermPositionVector[] getTermPositionVectors(int docNumber)

> throws IOException

> {

> return tvs.get(docNumber, true);

> }

>

Index: src/java/org/apache/lucene/index/SegmentTermDocs.java

===================================================================

RCS file: /home/cvspublic/jakarta-lucene/src/java/org/apache/lucene/index/SegmentTermDocs.java,v

retrieving revision 1.1.1.1

diff -w -r1.1.1.1 SegmentTermDocs.java

148a149,158

> }

>

>

> /\*\* Test if the enum has a current value. The enum is not valid until

> \* the first call to <code>next()</code>. After that, as long as <code>next()</code>

> \* returns <code>true</code>, the enum is valid. Once <code>next()</code> returns

> \* false, the enum again becomes invalid.

> \*/

> public final boolean isValid() {

> return freqCount != 0;

Index: src/java/org/apache/lucene/index/SegmentTermEnum.java

===================================================================

RCS file: /home/cvspublic/jakarta-lucene/src/java/org/apache/lucene/index/SegmentTermEnum.java,v

retrieving revision 1.2

diff -w -r1.2 SegmentTermEnum.java

183a184,201

>

> /\*\* Returns the current term's number. This is the number by which the term

> \* is known in termvectors obtained from this reader.

> \* Initially invalid, valid after next() called for the first time.\*/

> public final int termNumber() {

> return position;

> }

>

>

> /\*\* Test if the enum has a current value. The enum is not valid until

> \* the first call to <code>next()</code>. After that, as long as <code>next()</code>

> \* returns <code>true</code>, the enum is valid. Once <code>next()</code> returns

> \* false, the enum again becomes invalid.

> \*/

> public final boolean isValid() {

> return term != null;

> }

>

Index: src/java/org/apache/lucene/index/SegmentTermPositions.java

===================================================================

RCS file: /home/cvspublic/jakarta-lucene/src/java/org/apache/lucene/index/SegmentTermPositions.java,v

retrieving revision 1.1.1.1

diff -w -r1.1.1.1 SegmentTermPositions.java

112c112

< throw new RuntimeException();

---

> throw new RuntimeException("TermPositions does not support processing multiple documents in one call. Use TermDocs instead.");

Index: src/java/org/apache/lucene/index/SegmentsReader.java

===================================================================

RCS file: /home/cvspublic/jakarta-lucene/src/java/org/apache/lucene/index/SegmentsReader.java,v

retrieving revision 1.2

diff -w -r1.2 SegmentsReader.java

165a166,221

>

>

> // TermVector support

>

> /\*\* Return a term corresponding to the specified term number.

> \* If no term with this number exists, return null.

> \*/

> public Term getTerm(int termNumber) throws IOException {

> throw new RuntimeException("Unimplemented");

> }

>

> /\*\* Return an array of term frequency vectors for the specified document.

> \* The array contains a vector for each vectorized field in the document.

> \* Each vector vector contains term numbers and frequencies for all terms

> \* in a given vectorized field.

> \* If no such fields existed, the method returns null.

> \*/

> public TermFreqVector[] getTermFreqVectors(int docNumber)

> throws IOException {

> throw new RuntimeException("Unimplemented");

> }

>

>

> /\*\* Return a term frequency vector for the specified document and field. The vector

> \* returned contains term numbers and frequencies for all terms

> \* in the specified field of this document, if the field had storeTermVector flag set.

> \* If the flag was not set, the method returns null.

> \*/

> public TermFreqVector getTermFreqVector(int docNumber, String field)

> throws IOException

> {

> throw new RuntimeException("Unimplemented");

> }

>

> /\*\* Return a term position vector for the specified document. The vector

> \* returned is like a term frequency vector but also contains positions

> \* in which the terms were found in their fields.

> \*/

> public TermPositionVector getTermPositionVector(int docNumber, String field)

> throws IOException

> {

> throw new RuntimeException("Unimplemented");

> }

>

> /\*\* Return an array of term position vectors for the specified document.

> \* The array contains a vector for each vectorized field in the document.

> \* Each vector returned is like a term frequency vector but also contains positions

> \* in which the terms were found in their fields.

> \* If no such fields existed, the method returns null.

> \*/

> public TermPositionVector[] getTermPositionVectors(int docNumber)

> throws IOException

> {

> throw new RuntimeException("Unimplemented");

> }

>

232a289,296

>

> public final int termNumber() {

> throw new RuntimeException("Not supported.");

> }

>

> public final boolean isValid() {

> return term != null;

> }

241a306

> boolean valid = false;

259a325

> valid = true;

267c333,334

< } else

---

> } else {

> valid = false;

269a337

> }

312a381,384

> }

>

> public final boolean isValid() {

> return valid;

Index: src/java/org/apache/lucene/index/TermDocs.java

===================================================================

RCS file: /home/cvspublic/jakarta-lucene/src/java/org/apache/lucene/index/TermDocs.java,v

retrieving revision 1.1.1.1

diff -w -r1.1.1.1 TermDocs.java

105a106,113

> /\*\* Test if the enum has a current value. The enum is not valid until

> \* the first call to <code>next()</code>. After that, as long as <code>next()</code>

> \* returns <code>true</code>, the enum is valid. Once <code>next()</code> returns

> \* false, the enum again becomes invalid.

> \*/

> public boolean isValid();

>

>

Index: src/java/org/apache/lucene/index/TermEnum.java

===================================================================

RCS file: /home/cvspublic/jakarta-lucene/src/java/org/apache/lucene/index/TermEnum.java,v

retrieving revision 1.1.1.1

diff -w -r1.1.1.1 TermEnum.java

77a78,113

>

>

> // Term Vector support

> /\*\* Returns the current term's number. This is the number by which the term

> \* is known in termvectors obtained from this reader.

> \* Initially invalid, valid after next() called for the first time.

> \*/

> abstract public int termNumber();

>

> /\*\* Skips terms to the first beyond the current whose value is

> \* greater or equal to <i>target</i>. <p>Returns true iff there is such

> \* an entry. <p>Behaves as if written: <pre>

> \* public boolean skipTo(Term target) {

> \* do {

> \* if (!next())

> \* return false;

> \* } while (target > term());

> \* return true;

> \* }

> \* </pre>

> \* Some implementations are considerably more efficient than that.

> \*/

> public boolean skipTo(Term target) throws IOException {

> do {

> if (!next())

> return false;

> } while (target.compareTo(term()) > 0);

> return true;

> }

>

> /\*\* Test if the enum has a current value. The enum is not valid until

> \* the first call to <code>next()</code>. After that, as long as <code>next()</code>

> \* returns <code>true</code>, the enum is valid. Once <code>next()</code> returns

> \* false, the enum again becomes invalid.

> \*/

> abstract public boolean isValid();

Index: src/java/org/apache/lucene/index/TermInfosWriter.java

===================================================================

RCS file: /home/cvspublic/jakarta-lucene/src/java/org/apache/lucene/index/TermInfosWriter.java,v

retrieving revision 1.1.1.1

diff -w -r1.1.1.1 TermInfosWriter.java

125a126,132

> /\*\* Returns the position of the most recently added term.

> \* If no term has been added, returns -1.

> \*/

> final int position() {

> return size - 1;

> }

>

157a165,171

> }

>

> /\*\* Return the current size. This is also the next available term number that

> \* will be assigned to the next term that is passed to <code>add</code>.

> \*/

> final int size() {

> return size;

Index: src/java/org/apache/lucene/search/FilteredTermEnum.java

===================================================================

RCS file: /home/cvspublic/jakarta-lucene/src/java/org/apache/lucene/search/FilteredTermEnum.java,v

retrieving revision 1.1.1.1

diff -w -r1.1.1.1 FilteredTermEnum.java

85c85

< if (termCompare(term))

---

> if (termCompare(term)) {

86a87

> }

128a130,140

> }

>

>

> // Term Vector support

>

> public int termNumber() {

> return actualEnum.termNumber();

> }

>

> public boolean isValid() {

> return currentTerm == null || endEnum();

Index: src/java/org/apache/lucene/search/IndexSearcher.java

===================================================================

RCS file: /home/cvspublic/jakarta-lucene/src/java/org/apache/lucene/search/IndexSearcher.java,v

retrieving revision 1.2

diff -w -r1.2 IndexSearcher.java

90c90

< final int docFreq(Term term) throws IOException {

---

> public final int docFreq(Term term) throws IOException {

99c99

< final int maxDoc() throws IOException {

---

> public final int maxDoc() throws IOException {

Index: src/java/org/apache/lucene/search/MultiSearcher.java

===================================================================

RCS file: /home/cvspublic/jakarta-lucene/src/java/org/apache/lucene/search/MultiSearcher.java,v

retrieving revision 1.2

diff -w -r1.2 MultiSearcher.java

88c88

< final int docFreq(Term term) throws IOException {

---

> public final int docFreq(Term term) throws IOException {

121c121

< final int maxDoc() throws IOException {

---

> public final int maxDoc() throws IOException {

185a186,203

>

>

> /\*\* Return the number of documents that match the specified query with

> \* the score greater than the specified <code>minScore</code>. Set the

> \* <code>minScore</code> to <code>0.0</code> to include all matching documents.

> \* If <code>filter</code> is not null, only documents selected by the

> \* filter are considered.

> \*/

> public final int count(Query query, Filter filter, final float minScore)

> throws IOException

> {

> int totalCount = 0;

> for (int i = 0; i<searchers.length; i++) {

> totalCount += searchers[i].count(query, filter, minScore);

> }

> return totalCount;

> }

>

Index: src/java/org/apache/lucene/search/Searcher.java

===================================================================

RCS file: /home/cvspublic/jakarta-lucene/src/java/org/apache/lucene/search/Searcher.java,v

retrieving revision 1.2

diff -w -r1.2 Searcher.java

59a60

> import org.apache.lucene.util.BitVector;

77a79,103

> /\*\* Return the number of documents that match the specified query. \*/

> public final int count(Query query) throws IOException {

> return count(query, (Filter) null, 0.0F);

> }

>

> /\*\* Return the number of documents that match the specified query with

> \* the score greater than the specified <code>minScore</code>. Set the

> \* <code>minScore</code> to <code>0.0</code> to include all matching documents.

> \* If <code>filter</code> is not null, only documents selected by the

> \* filter are considered.

> \*/

> public int count(Query query, Filter filter, final float minScore)

> throws IOException

> {

> final BitVector results = new BitVector(maxDoc());

> search(query, filter, new HitCollector() {

> public void collect(int doc, float score) {

> if (score > minScore) results.set(doc);

> }

> });

> return results.count();

> }

>

>

>

112,113c138,146

< abstract int docFreq(Term term) throws IOException;

< abstract int maxDoc() throws IOException;

---

> /\*\* Return the number of documents in this searcher that contain specified term. \*/

> abstract public int docFreq(Term term) throws IOException;

>

> /\*\* Return a number that is 1 + maximum number of documents that can be found

> \* by this searcher.

> \*/

> public abstract int maxDoc() throws IOException;

>

>

Index: src/java/org/apache/lucene/store/InputStream.java

===================================================================

RCS file: /home/cvspublic/jakarta-lucene/src/java/org/apache/lucene/store/InputStream.java,v

retrieving revision 1.1.1.1

diff -w -r1.1.1.1 InputStream.java

63,64d62

<

< /\*\* A random-access input stream \*/

83a82

> /\*\* InputStream-like methods @see java.io.InputStream \*/

99a99,102

> /\*\* Read an integer from the stream. The integer must have been written

> \* by a call to OutputStream.writeInt. It is stored as four bytes, from most to least

> \* significant.

> \*/

104a108,116

> /\*\* Read a compressed (or Variable length) integer from the stream. The

> \* integer must have been written by a call to OutputStream.writeVInt.

> \* It is stored as a series of bytes, from least significant to the most

> \* significant. Each byte contains 7 bits of data and the 8th (0x80) bit

> \* that indicates that there are more bytes to be read for this integer.

> \* With this format, smaller integers occupy only one byte,

> \* larger ones - two bytes, and so on up to 5 bytes. Note that negative numbers

> \* are always stored using the maximum number of bytes.

> \*/

114a127,130

> /\*\* Read a long from the stream. The long must have been written by a call to

> \* OutputStream.writeLong. It is stored as 8 bytes, from most significant to the least

> \* significant.

> \*/

118a135,140

> /\*\* Read a compressed (or Variable length) long from the stream. The long must

> \* have been written by a call to OutputStream.writeVLong. It is stored

> \* similarly to the VInt, but may occupy 1 to 10 bytes. Note that negative numbers

> \* are always stored using the maximum number of bytes.

> \*

> \*/

128a151,154

> /\*\* Read a string from the stream. The string must have been written by a call

> \* to OutputStream.writeString. It is stored as a VInt (see readVInt)

> \* indicating the string size, followed by that many chars (see readChars).

> \*/

136a163,169

> /\*\* Read an array of characters, placing them into the provided buffer.

> \* The read characters are placed into array starting with the index <i>start</i>

> \* and continuing for <i>length</i> characters. The characters must have been

> \* written with a call to OutputStream.writeChards. Each character is stored

> \* using one, two, or three bytes, depending on the value of the character.

> \* This represents a standard encoding of Unicode characters known as UTF-8.

> \*/

181a215

> /\*\* RandomAccessFile-like methods @see java.io.RandomAccessFile \*/

193a228

> /\*\* RandomAccessFile-like methods @see java.io.RandomAccessFile \*/

197a233,237

> /\*\* Create a clone of this stream. The clone provides access to the same

> \* undelying descriptor as the original file, however it maintains its own

> \* buffer and file position so it can be used concurrently with the original

> \* file and other clones.

> \*/

Index: src/java/org/apache/lucene/store/OutputStream.java

===================================================================

RCS file: /home/cvspublic/jakarta-lucene/src/java/org/apache/lucene/store/OutputStream.java,v

retrieving revision 1.1.1.1

diff -w -r1.1.1.1 OutputStream.java

63,64d62

<

< /\*\* A random-access output stream \*/

78a77

> /\*\* OutputStream-like methods @see java.io.InputStream \*/

83a83,85

> /\*\* Write an integer into the stream. The integer can be read by calling

> \* InputStream.readInt. It is stored using four bytes.

> \*/

90a93,98

> /\*\* Write a compressed (or Variable length) integer into the stream.

> \* The integer can be read by calling InputStream.readVInt.

> \* It is stored using from 1 to 5 bytes, depending on the value of the

> \* integer. Note that negative numbers

> \* are always stored using the maximum number of bytes.

> \*/

98a107,109

> /\*\* Write a long into the stream. The long can be read by calling InputStream.readLong.

> \* It is stored using 8 bytes.

> \*/

103a115,119

> /\*\* Write a compressed (or Variable length) long into the stream. The long can

> \* be read by calling InputStream.readVLong. It is stored using from 1 to

> \* 10 bytes depending on the value of the long. Note that negative numbers

> \* are always stored using the maximum number of bytes.

> \*/

111a128,131

> /\*\* Write a string into the stream. The string can be read by calling

> \* InputStream.readString. It is stored as a VInt representing the number of

> \* characters, followed by that many characters (see writeChars).

> \*/

117a138,142

> /\*\* Write an array of characters into the stream. The array can be read by

> \* calling InputStream.readChars. Each character is stored using from one to

> \* three bytes depending on the value of the character.

> \* This represents a standard encoding of Unicode characters known as UTF-8.

> \*/

143a169

> /\*\* Flush and close the stream. \*/

152a179

> /\*\* RandomAccessFile-like methods @see java.io.RandomAccessFile \*/

157a185

> /\*\* RandomAccessFile-like methods @see java.io.RandomAccessFile \*/

--------------010909000801060505060408--