http://mail-archives.apache.org/mod\_mbox/lucene-dev/200110.mbox/%3c3BD074CF.3070106@earthlink.net%3e

From lucene-dev-return-260-apmail-jakarta-lucene-dev-archive=jakarta.apache.org@jakarta.apache.org Fri Oct 19 18:45:30 2001

Return-Path: <lucene-dev-return-260-apmail-jakarta-lucene-dev-archive=jakarta.apache.org@jakarta.apache.org>

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

Received: (qmail 6050 invoked by uid 500); 19 Oct 2001 18:45:30 -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 5982 invoked from network); 19 Oct 2001 18:45:28 -0000

Message-ID: <3BD074CF.3070106@earthlink.net>

Date: Fri, 19 Oct 2001 12:45:35 -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: Sorting vs. document boosting

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

Content-Transfer-Encoding: 7bit

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

With the addition of term vectors, I think there are new ways that we

can now do sorting in Lucene. It's different from boosting documents,

but sometimes it is specifically sorting that is required. I've written

HitCollectors that retrieve term vectors on each call to collect() and

the performance remains quite acceptable (that's even without further

optimizations to the term vectors). So it is now possible to write

HitCollectors that retrieve termvectors for a specific field and sort

documents they collect by values in those fields. If the field is a

"keyword" it will only have a single term, which is probably most

appropriate for sorting. The term numbers used by the term vectors are

designed to be ordered by lexicographical ordering of the terms

themselves. In other words, if number of term A is less than number of

term B, then term A is also less than the term B. If the term text is

choosen with thought, this will allow very efficient sorting, I think.

Perhaps sorting should be a property that is set on a query? It could be

as simple as query.sortBy(String field, boolean ascending). The queyr

will then sort by the specified field's first term. The field must be

vectorized. This way it could influence the behavior of the default

HitCollector (which now sorts only by score).