http://mail-archives.apache.org/mod\_mbox/lucene-dev/200110.mbox/%3c3BC7F246.2030204@earthlink.net%3e

From dmitrys@earthlink.net Sat Oct 13 07:50:36 2001

Return-Path: <dmitrys@earthlink.net>

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

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

Received: (qmail 26171 invoked from network); 13 Oct 2001 07:50:36 -0000

Received: from relay4.uswest.net (HELO relay2.uswest.net) (63.226.138.12)

by daedalus.apache.org with SMTP; 13 Oct 2001 07:50:36 -0000

Received: (qmail 41964 invoked by uid 0); 13 Oct 2001 07:50:44 -0000

Received: from unknown (HELO earthlink.net) (65.100.117.194)

by relay2.uswest.net with SMTP; 13 Oct 2001 07:50:44 -0000

Message-ID: <3BC7F246.2030204@earthlink.net>

Date: Sat, 13 Oct 2001 01:50:30 -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 retrieval implementation questions

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

Greetings,

I have to apologize for so many messages to the list, but I really have

to get the TermVector stuff working within the next few days because the

next release of our application is going to depend on it. Once the

release is done, it will be much harder (for us) to make changes to file

formats. So I'm going to continue being a bit noisy for a while, just so

everyone has an opportunity to comment on the changes as I'm making

them, and so we don't have to make too many changes later on. With

enough input, the result will be a whole new set of capabilities for

Lucene that everyone can use! Isn't that cool?

Ok, that said, here are a few questions to those in the know:

\*) Is there any particular reason why the "tokenized" bit is stored in

the fdt file, while the "indexed" bit is stored in the fnm? Why not put

both in fnm? I'm not proposing that we do this (compatibility must be

preserved), but I'm just making sure I find the right place for the

other bits that I'm planning to add.

\*) I'm planning to add another bit: "storeTermVector" (better name,

anyone?), which will indicate that the field's term vector will need to

be stored.

\*) The term vector, as I understand it, is a list of unique terms that

occur in a given field. They will be stored by term id (in ascending

order of IDs, not terms). In addition to the terms, I'm planning to

store the frequency of the term (the number of times it occurs in the

field). This, together with the total number of terms in the field,

should be enough to compute the term's weight, right? My application

doesn't need these weights, so I'm not sure what people need in this

regard. Please advise.

\*) In addition to the terms and frequencies, I will also store positions

in which these terms occur in the field. Actually, this is already

stored (used by the TermPositions functionality), so I will only store

pointers into the prx file. This may not be needed for clustering, but I

need this for my application. Some of the text processing that we do is

based on relative positioning of terms in a document.

\*) Between the term vector and the positions, it will be possible to

recreate the contents of a field except for word breaks, so I considered

using the "stored" + "tokenized" to mean that a termvector should be

stored and only storing the information in this way, instead of

essentially storing it twice. However, at present, I think that it is

useful to store the original content, breaks and all. Reactions,

suggestions?

\*) Speaking of the stored fields, someone suggested adding binary

storage to documents so that serialized objects can be stored. From what

I can see, it would be pretty easy to define a new field type that

stores binary data, add a flag into the bits stored in fdt file for this

field, and then write it out as an array of bytes instead of a String.

This could be useful for my application as well, although currently I

have a workaround so this is not required. Any votes for or against

adding this feature?

\*) Preliminary file structures. These are the files I'm planning to add

to each segment:

"fvx" file - Field Vector Index. Modeled on the fdx file. Has a

fixed length, 8-byte, record per document in a segment. The 8 bytes

store a long pointer into the "fvt" file where the record for this

document begins.

"fvt" file - Field Vector Table. Modeled in part on "fdt" and in

part on "tis" file. Each document record in this file looks like this:

document\_record :

[VInt] - number of fields (only fields with storeTermVector flag set)

{ field\_record, ... }- field records, as many as specified above

field\_record :

[VInt] - field number, just like in the "fdt" file

[byte] - flags, don't know if we are going to need any, but seems

like we might?

[VInt] - maxTerm, 1+numberOfTerms - just like maxDoc. Used for

array allocation and term weight calculations?

[VInt] - numTerms, count of unique terms in the vector, number of

term records that follow

{ term\_record, ... } - term records, as many as specified above,

represent unique terms in the field

term\_record:

[VInt] - term id increment, restarts from 0 for each field

[VInt] - term frequency in this field, used for weight

calculations and for the count of positions in "prx" file

[VInt] - "prx" pointer increment, restarts from 0 for each field

A couple of questions on the file formats that I would really like

feedback on:

\*) First: am I setting myself up in any way? Meaning, does this design

have inherent limitations that will cause things to slow down or be

awkward to implement?

\*) Specifically, I'm trying to identify what makes access to the

document fields ("fdx" and "fdt" files) slow, and make sure I avoid

those problems. From what I can tell, the only thing that makes that

access slow is the size of the document data, in which case we have

nothing to worry about. Is that right?

\*) I don't see any place to apply the trick used in the "tii" and "tis"

files - namely loading every 128th element into memory and using that as

an index into a larger file. I don't think this can be applied because

we are really not "searching" for anything, we just do direct access by

document id. Am I missing anything?

\*) Finally, users are likely to access termvectors from a given field

only. This may be a good reason to optimize access to each field\_record

in the proposed "fvt" file. I can see two ways of doing this:

1 - include a field record jump table in the beginning of each

document's record in the fvt file. The table would include pointer

increments for each of the fields. Only this table will need to be read

and then a reader can jump directly to that field's term vector. This

may be hard to write because I will need to seek the writer stream back

when the values for the table are known. Hm... This problem might just

kill this idea right there...

2 - also include a field record jump table, but the values would be

pointer increments into a different file that will only contain field

records. This means that yet another file will need to be opened and

read. But it may not be such a big deal.

So the question is: which way would be preferable, and are there other

ways that might be even better?

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

Issues still to be addressed:

- finalize public API for indexing and for access to this data

- exact classes that will be responsible for reading and writing this data

- cross-segment term id merge strategy for queries (if any)

- cross-searcher term id merge strategy for queries on MultiSearcher (if

any)

- backward compatibility. Ideally, a given index should be able to

operate with some old and some new segments.

- translation from stemmed form to original form or a term for display

purposes

- consider implementing "termSearch" method on Searcher, which would

provide framework for executing queries that result in selection of

terms rather then selection of documents.

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

Thanks everyone for making Lucene so great!

Let's make it even better! :)

-dmitry