Text/buffer representation

Arguably, wrangling and displaying text are the most important things a text editor does. Vim represents text as a sequence of lines (see `mem{line,file}.{c,h}`, needs extra verification).

Operations on lines are thus easy and efficient. It seems to be a good fit for the many operations Vim supports (undo/redo, text operators, ...). Even operations that are not necessarily line-based, like block mode operations, are easy to implement over a line-based API.

Yet it is always instructive to learn about how other text editors tackle the issue.

One of the disadvantages of the line-based approach seems to be that Vim has problems efficiently handling files that have no/little newlines. Think of large XML files that curiously have no newlines, it is often slow. Syntax highlighting is likely to blame in this case. More research is necessary to discover if this is can be remedied within the line-based structure, or if a different approach is necessary.

Some alternative approaches:

Piece Buffer

As described in "Data Structures for Text Sequences" comments, pdf. Notably implemented in the vis text editor, which supports many Vim-like operators.

Advantages:

- Can `mmap()` the file into memory. This is the best way to seamlessly support editing huge files, especially on OS'es that do demand paging (all Neovim supported platforms do that).

Disadvantages:

- As noted by the **vis** author, text encoding conversion undoes the `mmap()` advantage mentioned above. A workaround might be possible, but it would really complicate the implementation. One idea for making it work is to make an overlay, see neovim/#1767.

Rope

See wikipedia. TODO: more discussion on how it could be applied to Neovim.

Gap Buffer

See wikipedia. TODO: more discussion on how it could be applied to Neovim.