4 Index construction

In this chapter, we look at how to construct an inverted index. We call this INDEXING process index construction or indexing; the process or machine that performs it INDEXER the indexer. The design of indexing algorithms is governed by hardware constraints. We therefore begin this chapter with a review of the basics of computer hardware that are relevant for indexing. We then introduce blocked

sort-based indexing (Section 4.2), an efficient single-machine algorithm designed for static collections that can be viewed as a more scalable version of the basic sort-based indexing algorithm we introduced in Chapter 1. Section 4.3 describes single-pass in-memory indexing, an algorithm that has even better scaling properties because it does not hold the vocabulary in memory. For very large collections like the web, indexing has to be distributed over computer clusters with hundreds or thousands of machines.

We discuss this in Section 4.4. Collections with frequent changes require dynamic indexing introduced in Section 4.5 so that changes in the collection are immediately reflected in the index. Finally, we cover some complicating issues that can arise in indexing – such as security and indexes for ranked

retrieval – in Section 4.6.

Index construction interacts with several topics covered in other chapters.The indexer needs raw text, but documents are encoded in many ways (see Chapter 2). Indexers compress and decompress intermediate files and the final index (see Chapter 5). In web search, documents are not on a local file system, but have to be spidered or crawled (see Chapter 20). In enterprise search, most documents are encapsulated in varied content management systems, email applications, and databases. We give some examples in Section 4.7. Although most of these applications can be accessed via http, native Application Programming Interfaces (APIs) are usually more efficient. The reader should be aware that building the subsystem that feeds raw text to the indexing process can in itself be a challenging problem.

◭ Table 4.1 Typical system parameters in 2007. The seek time is the time needed to position the disk head in a new position. The transfer time per byte is the rate of transfer from disk to memory when the head is in the right position.

4.1 Hardware basics

When building an information retrieval (IR) system, many decisions are based on the characteristics of the computer hardware on which the system runs. We therefore begin this chapter with a brief review of computer hardware. Performance characteristics typical of systems in 2007 are shown in Table 4.1. A list of hardware basics that we need in this book to motivate IR system design follows.

• Access to data in memory is much faster than access to data on disk. It takes a few clock cycles (perhaps 5×10^{-9} seconds) to access a byte in memory, but much longer to transfer it from disk (about 2×10^{-8} seconds). Consequently, we want to keep as much data as possible in memory, especially those data that we need to access frequently. We call the CACHING technique of keeping frequently used disk data in main memory caching.

• When doing a disk read or write, it takes a while for the disk head to move to the part of the disk where the data are located. This time is called SEEK TIME the seek time and it averages 5 ms for typical disks. No data are being transferred during the seek. To maximize data transfer rates, chunks of data

that will be read together should therefore be stored contiguously on disk. For example, using the numbers in Table 4.1 it may take as little as 0.2 seconds to transfer 10 megabytes (MB) from disk to memory if it is stored as one chunk, but up to $0.2 + 100 \times (5 \times 10{-}3) = 0.7$ seconds if it is stored in 100 noncontiguous chunks because we need to move the disk head up to 100 times.

• Operating systems generally read and write entire blocks. Thus, reading a single byte from disk can take as much time as reading the entire block.

Block sizes of 8, 16, 32, and 64 kilobytes (KB) are common. We call the part BUFFER of main memory where a block being read or written is stored a buffer.

• Data transfers from disk to memory are handled by the system bus, not by the processor. This means that the processor is available to process data during disk I/O. We can exploit this fact to speed up data transfers by storing compressed data on disk. Assuming an efficient decompression algorithm, the total time of reading and then decompressing compressed data is usually less than reading uncompressed data.

• Servers used in IR systems typically have several gigabytes (GB) of main memory, sometimes tens of GB. Available disk space is several orders of magnitude larger.