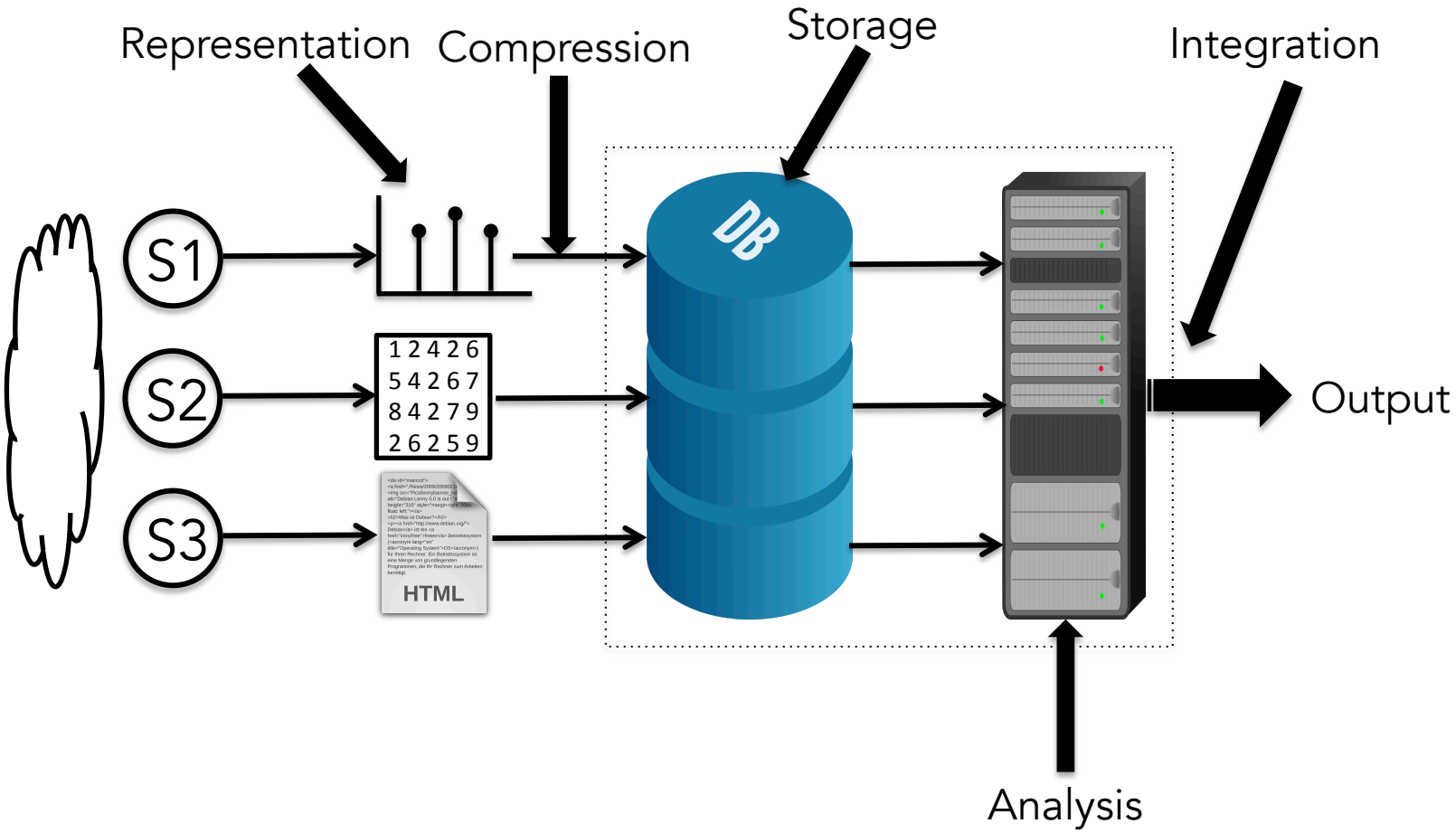
Week 12

Data Compression

Reference and Slide Source: ChengXiang Zhai and Sean Massung. 2016. Text Data Management and Analysis: a Practical Introduction to Information Retrieval and Text Mining. Association for Computing Machinery and Morgan & Claypool, New York, NY, USA.

What is Multimedia?

Multiple Carriers of information


Representation
Compression
Storage
Integration
S1
S2
S3
1 2 4 2 6
5 4 2 6 7
8 4 2 7 9
2 6 2 5 9
HTML
DB
Output
Analysis

Already Discussed Lossless Techniques

- Predictive coding
- Run length coding
- Huffman coding

Already Discussed Lossy Techniques

Quantization

Quantization is at the core of all lossy quantization techniques!

E.g. JPEG, MPEG, MP3, DPCM

How to Compress Text?

The computer science and engineering department offers many computer related courses. Although there is science term in the name, no science courses are offered in the department.

Differential Coding?

The computer science and engineering department offers many computer related courses. Although there is science term in the name, no science courses are offered in the department.

Not a good idea!

Run Length Coding?

The computer science and engineering department offers many computer related courses. Although there is science term in the name, no science courses are offered in the department.

Not a good idea!

Huffman Coding?

The computer science and engineering department offers many computer related courses. Although there is science term in the name, no science courses are offered in the department.

Huffman Coding

- Removes the source information
- Frequent symbols assigned shorter codes
- Prefix coding

Can we refer to a dictionary entry of the word?

Dictionary-based Compression

- Do not encode single symbols as variable-length bit strings
- Encode variable-length string of symbols as single token
- The tokens form an index into a phrase dictionary
- If tokens are smaller than phrases, we have compression

Lempel-Ziv-Welsh (LZW) Coding

- A dictionary-based coding algorithm
- Build the dictionary dynamically
- Initially the dictionary contains only character codes
- The remaining entries in the dictionary are then build dynamically

Lempel–Ziv–Welch (LZW) Compression

```
set w = NIL
    loop
        read a character k
        if wk exists in the dictionary
            w = wk
        else
            output the code for w
            add wk to the dictionary
            w=k
    endloop
```

Ref: http://users.ece.utexas.edu/~ryerraballi/MSB/pdfs/M2L3.pdf

Example

Input String: ^WED^WE^WEE^WEB^WET

w	k	Output	Index	Symbol
NIL	^			
^	W	^	256	^W
W	E	W	257	WE
E	D	E	258	ED
D	^	D	259	D^
^	W			
^W	E	256	260	^WE
E	^	E	261	E^
^	W			
^W	E			
^WE	E	260	262	^WEE
E	^			
E^	W	261	263	E^W
W	E			
WE	B	257	264	WEB
B	^	B	265	B^
^	W			
^W	E			
^WE	T	260	266	^WET
T	EOF	T		

```
set w = NIL
 loop
        read a character k
        if wk exists in the dictionary
                w = wk
        else
                output the code for w
                add wk to the dictionary
                w = k
endloop
```

Ref: http://users.ece.utexas.edu/~ryerraballi/MSB/pdfs/M2L3.pdf

LZW Decompression

read fixed length token k (code or char)

output k
w=k
loop

    read a fixed length token k

    entry = dictionary entry for k

    output entry
    add w + first char of entry to the dictionary

    w = entry

endloop

Ref: http://users.ece.utexas.edu/~ryerraballi/MSB/pdfs/M2L3.pdf

Example

Input String (to decode): ^WED<256>E<260><261><257>B<260>T

w	k	Output	Index	Symbol
	^	^		
^	W	W	256	^W
W	E	E	257	WE
E	D	D	258	ED
D	<256>	^W	259	D^
^W	E	E	260	^WE
E	<260>	^WE	261	E^
^WE	<261>	E^	262	^WEE
E^	<257>	WE	263	E^W
WE	B	B	264	WEB
B	<260>	^WE	265	B^
^WE	T	T	266	^WET

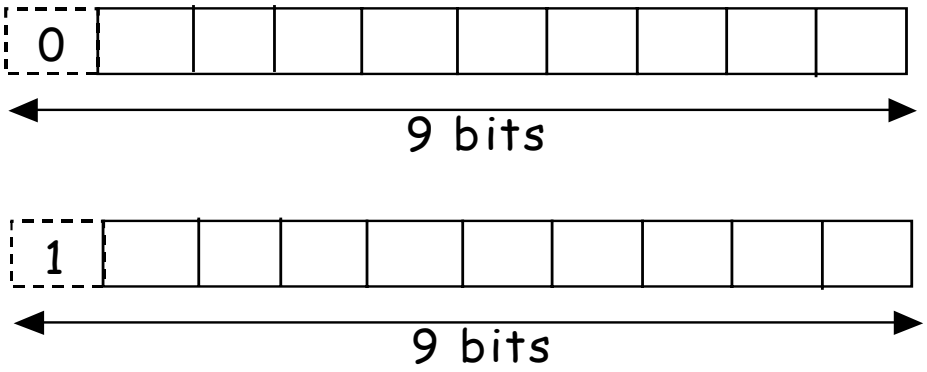
```
read a fixed length token k
      (code or char)
output k
w = k
loop
    read a fixed length token k
        (code or char)
    entry = dictionary entry for k
    output entry
    add w + first char of entry to
        the dictionary
    w = entry
endloop
```

Ref: http://users.ece.utexas.edu/~ryerraballi/MSB/pdfs/M2L3.pdf

Where is Compression?

- Input String: ^WED^WE^WEE^WEB^WET
  - 19 * 8 bits = 152 bits
- Encoded: ^WED<256>E<260><261><257>B<260>T
  - 12*9 bits = 108 bits (7 symbols and 5 codes, each of 9 bits)
- Why 9 bits?

Ref: http://users.ece.utexas.edu/~ryerraballi/MSB/pdfs/M2L3.pdf

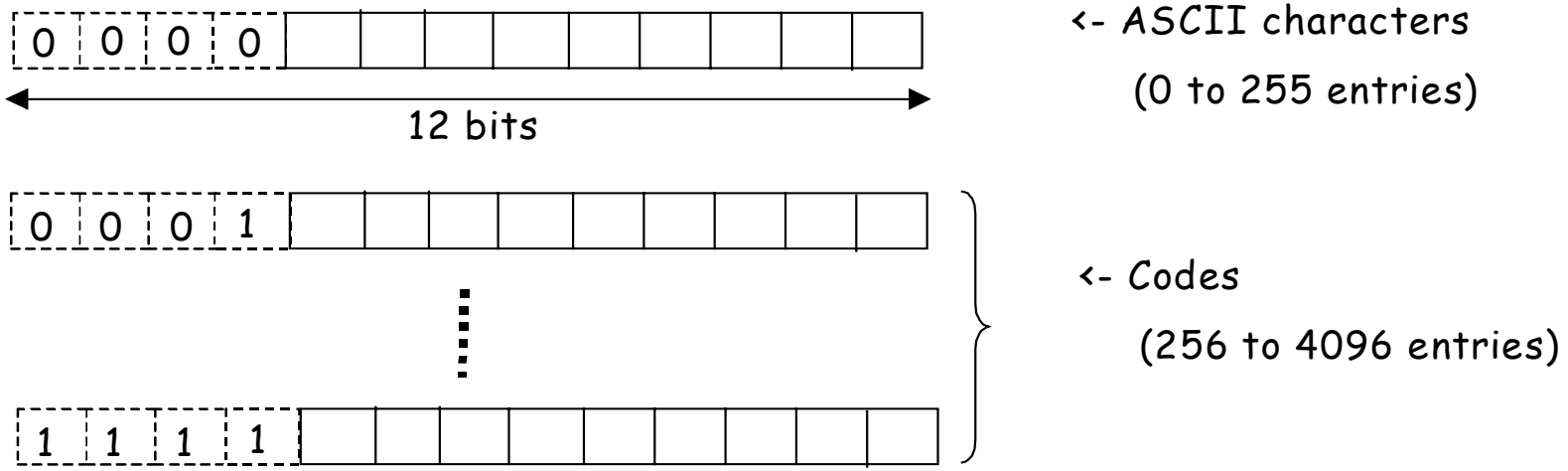


<- ASCII characters
(0 to 255)

<- Codes
(256 to 512)

Ref: http://users.ece.utexas.edu/~ryerraballi/MSB/pdfs/M2L3.pdf

Original LZW Uses 12-bit Codes!



Ref: http://users.ece.utexas.edu/~ryerraballi/MSB/pdfs/M2L3.pdf

What if we run out of codes?

- Flush the dictionary periodically
- Or Grow length of codes dynamically