Joint Source-Channel LZ'77 Coding

Stefano Lonardi
University of California, Riverside

Wojciech Szpankowski
Purdue University, West Lafayette

Source vs. Channel coding

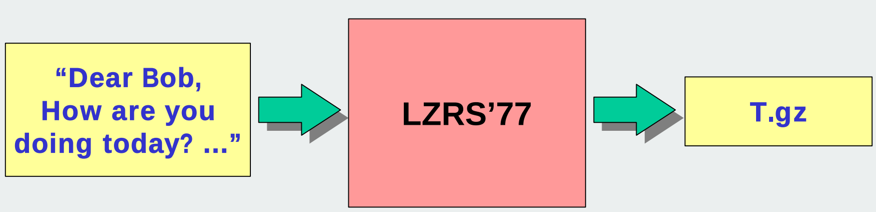
- Source coding: represent the source information with the minimum of symbols
- Channel coding: represent the source information in a manner that minimizes the error probability in decoding

Problem definition

- How to achieve joint source and channel coding in LZ'77 (by adding error resiliency)
  - without significantly degrading the compression performance,
  - and keeping backward compatibility with the original LZ'77?

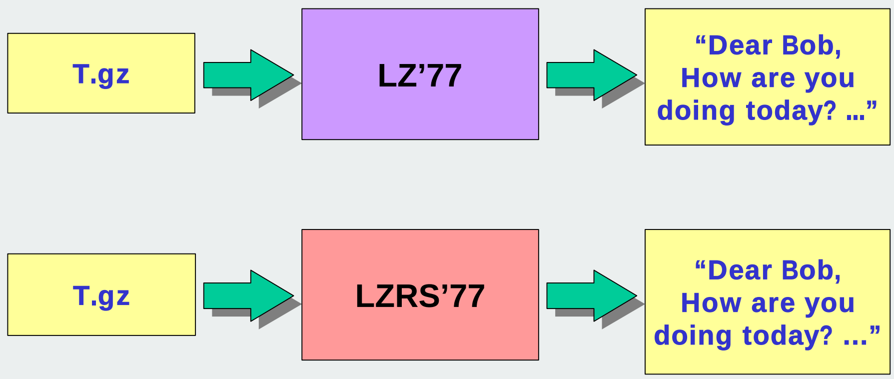
Stefano Lonardi
Department of CS and E
Bourns College of Engineering
University of California, Riverside

Encoding



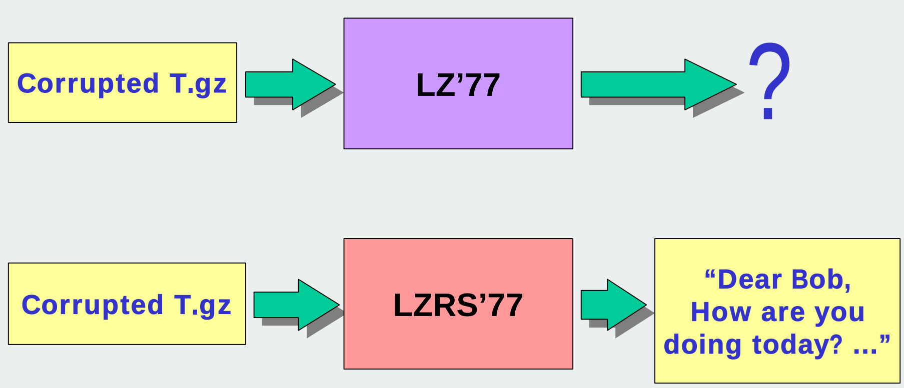
Stefano Lonardi
Department of CS and E
Bourns College of Engineering
University of California, Riverside

Decoding (no errors)



Stefano Lonardi
Department of CS and E
Bourns College of Engineering
University of California, Riverside

Decoding (with errors)



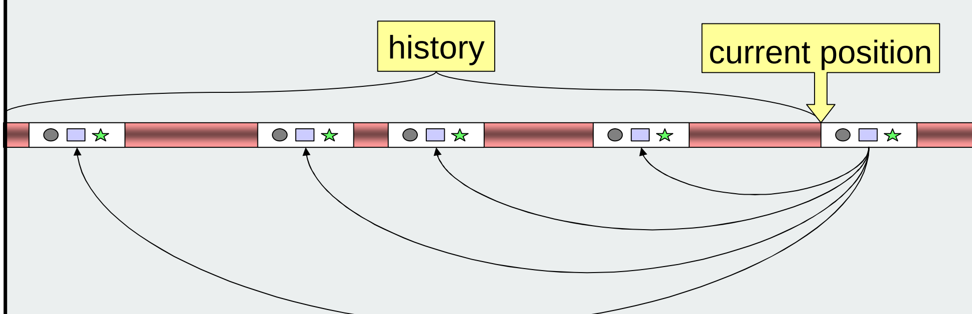
Stefano Lonardi
Department of CS and E
Bourns College of Engineering
University of California, Riverside

Roadmap

- We will show how to obtain extra redundant bits from LZ'77
- We will show how to achieve error resiliency in LZ'77

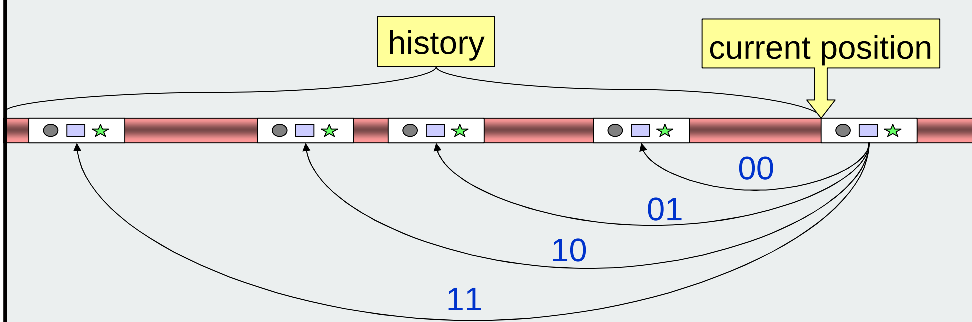
Stefano Lonardi
Department of CS and E
Bourns College of Engineering
University of California, Riverside

LZ'77: which of these pointers do we choose?



Stefano Lonardi
Department of CS and E
Bourns College of Engineering
University of California, Riverside

By choosing one of these pointers we are recovering two extra redundant bits. Note that we are not changing LZ'77



Stefano Lonardi
Department of CS and E
Bourns College of Engineering
University of California, Riverside

Extra bits recovering

- Definition: a LZ'77 phrase has multiplicity *q* if has exactly *q* matches in the history
- Given a phrase with multiplicity *q*, we can recover $\lfloor \log_2 q \rfloor$ bits

Stefano Lonardi
Department of CS and E
Bourns College of Engineering
University of California, Riverside

Average case analysis

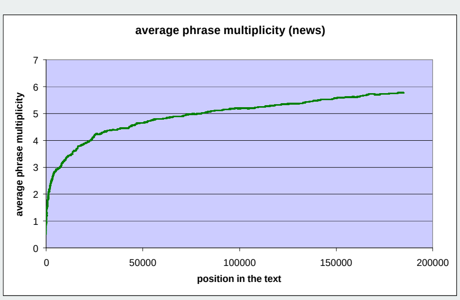
- Theorem: Let Q_n be the random variable associated with the multiplicity q of a phrase in a string of length n. For a Markov source

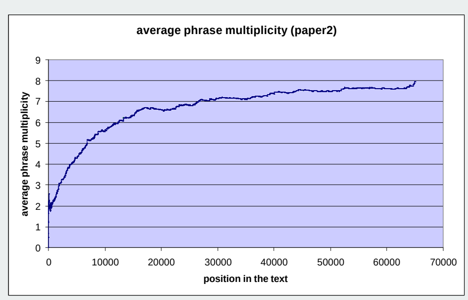
$$E[Q_n]=O(1)$$

as *n? 8*

Stefano Lonardi
Department of CS and E
Bourns College of Engineering
University of California, Riverside

Average phrase multiplicity





Stefano Lonardi
Department of CS and E
Bourns College of Engineering
University of California, Riverside

Recent results

- Theorem: For memoryless sources

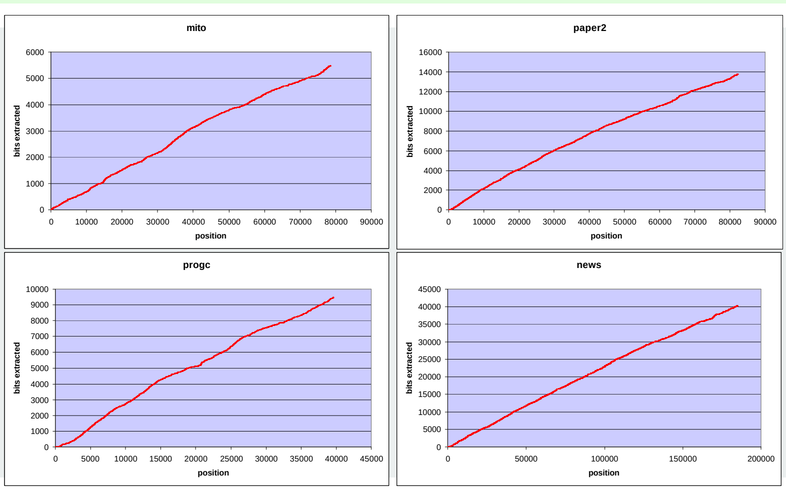
$$E[Q_n] = \frac{1}{H} + \text{ small fluctuations}$$

$$P[Q_n = k] = \frac{p(1-p)^k + (1-p)p^k}{kH}$$

where H is the entropy of the source, and p is the probability of generating a "0"

Stefano Lonardi
Department of CS and E
Bourns College of Engineering
University of California, Riverside

Number of bits recovered



Remark: more bits can be recovered by relaxing the greediness

Stefano Lonardi
Department of CS and E
Bourns College of Engineering
University of California, Riverside

Reed Solomon codes

- RS codes are block-based error correcting codes (BCH family)
- *RS(a,b)* code
  - $a=2^s-1$, where *s* is the datum size
  - has *(a-b)* "parity" bits
  - can correct up to *(a-b)/2* errors
- We used *RS(255,255-2e),* which can correct up to *e* errors

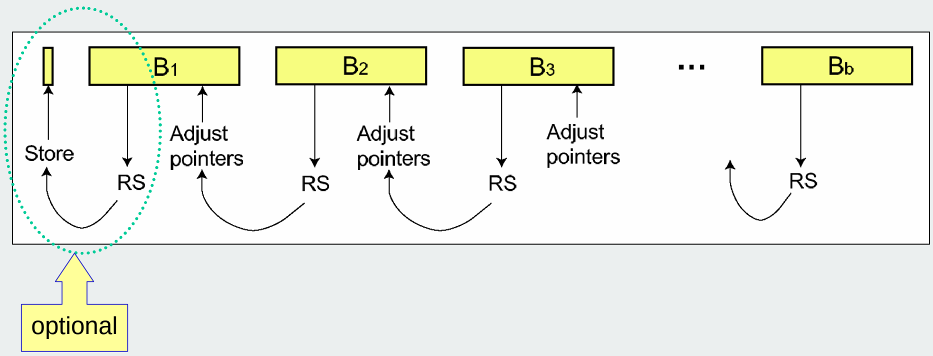
Stefano Lonardi
Department of CS and E
Bourns College of Engineering
University of California, Riverside

LZRS'77 encoder (off-line)

- **compress** the file with LZ'77
- **break** the compressed file in blocks $B_1,\ldots, B_m$ of size *255-2e*
- **for** $i \leftarrow m$ **downto** *2*
  - **encode** with *RS(255,255-2e)* block B_i
  - **embed** the extra *2e* parity bits in the pointers of block B_{i-1}
- **encode** with *RS(255,255-2e)* block B_1
- **store** the extra parity bits at the beginning of the file

Stefano Lonardi
Department of CS and E
Bourns College of Engineering
University of California, Riverside

LZRS'77 encoding



Stefano Lonardi
Department of CS and E
Bourns College of Engineering
University of California, Riverside

LZRS'77 decoder (on-line)

- (assume RS_i are the *2e* parity bits for B_i)
- **decode** and **correct** block B_1+RS_1
- **decompress** block B_1 and **recover** RS_2
- **for** $i \leftarrow 2$ **to** m
  - **decode** and **correct** block B_i+RS_i
  - **decompress** block B_i and **recover** RS_{i+1}

Stefano Lonardi
Department of CS and E
Bourns College of Engineering
University of California, Riverside

Experiments: **`gzip`**

- **`gzip`** issues pointers in a sliding window of 32Kbytes (typically)
- The length of phrases is represented by 8 bits (3-258)
- Strings smaller than 3 symbols are encoded as literals

Stefano Lonardi
Department of CS and E
Bourns College of Engineering
University of California, Riverside

`gzip`

- **`gzip`** always chooses the most "recent" occurrence of the longest prefix

  "…the hash chains are searched starting from the most recent strings, to favor small distances and thus take advantage of the Huffman coding…"

Stefano Lonardi
Department of CS and E
Bourns College of Engineering
University of California, Riverside

"Hacking" **gzip**

- We modified **gzip-1.2.4** to evaluate the potential degradation of compression performance due to changing the rule of choosing always the most "recent" occurrence
- As a preliminary experiment, we simply chose one pointer at random

Stefano Lonardi
Department of CS and E
Bourns College of Engineering
University of California, Riverside

gzip vs. **gzipS**

file size	gzip	gzipS	*file*	*bytes embedded*
111,261	39,473	39,511	bib	1,721
768,771	333,776	336,256	book1	14,524
610,856	228,321	228,242	book2	10,361
102,400	69,478	71,168	geo	4,101
377,109	155,290	156,150	news	5,956
21,504	10,584	10,783	obj1	353
246,814	89,467	89,757	obj2	3,628
53,161	20,110	20,204	paper1	937
82,199	32,529	32,507	paper2	1,551
46,526	19,450	19,567	paper3	893
13,286	5,853	5,898	paper4	249
11,954	5,252	5,294	paper5	210
38,105	14,433	14,506	paper6	738
513,216	62,357	61,259	pic	3,025
39,611	14,510	14,660	progc	736
71,646	18,310	18,407	progl	1,106
49,379	12,532	12,572	progp	741
93,695	22,178	22,098	trans	1,201

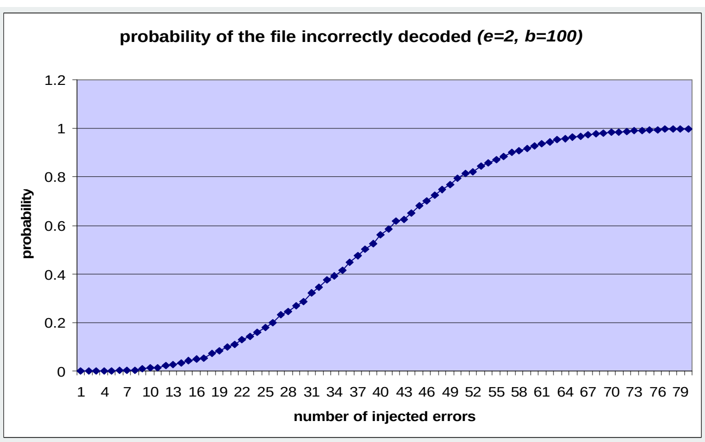
Stefano Lonardi
Department of CS and E
Bourns College of Engineering
University of California, Riverside

Error correction (simulation)

- We chose *e=1, e=2* and *b=10, b=100*
- For *b* blocks, we injected *1,…,b* uniformly distributed errors
- We measured the number of times that the file was decoded correctly (out of a few hundreds simulations)

Stefano Lonardi
Department of CS and E
Bourns College of Engineering
University of California, Riverside

Error-correction



Stefano Lonardi
Department of CS and E
Bourns College of Engineering
University of California, Riverside

Findings

- Method to recover extra redundant bits from LZ’77
- Extra bits allow to incorporate error resiliency in LZ’77
  - backward-compatible (deployment without disrupting service)
  - compression degradation due to the extra bits is almost negligible

Stefano Lonardi
Department of CS and E
Bourns College of Engineering
University of California, Riverside