Teoria da Informação

Algoritmos Lempel-Ziv: LZ77, LZ78 e LZW

Miguel Barão

ESCOLA DE CIÊNCIAS E TECNOLOGIA
DEPARTAMENTO DE INFORMÁTICA

Os algoritmos apresentados nestes slides foram originalmente propostos nos artigos seguintes:


Jacob Ziv, Abraham Lempel,
"A Universal Algorithm for Sequential Data Compression",
IEEE Transactions on Information Theory, 1977.

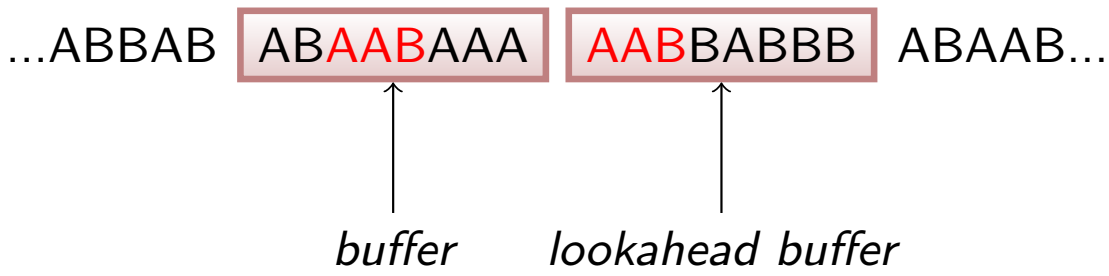

Jacob Ziv, Abraham Lempel,
"Compression of Individual Sequences via VariableRate Coding",
IEEE Transactions on Information Theory, 1978.


T. A. Welch,
"A Technique for High Performance Data Compression",
Computer, 1984.

Secção 1

Algoritmo Lempel-Ziv (1977)

- É um código universal (não depende da fonte).
- Codifica os segmentos futuros copiando de um buffer os segmentos gerados mais recentemente. Mais precisamente, procura a maior string no lookahead buffer que existe no buffer da esquerda:



- O output gerado pelo algoritmo consiste numa sequência de tuplas:

  (Ponteiro,Comprimento,NovoSímbolo)

  em que
  - Ponteiro indica o offset da string no buffer da esquerda;
  - Comprimento indica qual o comprimento da string a copiar do buffer;
  - NovoSímbolo indica qual o símbolo do lookahead buffer seguinte à string encontrada.

  No exemplo acima, se contarmos o offset a partir da direita, temos (6,3,B).

Exemplo

String: `AABCBBABC`.
Codificação: `(0,0,A) (1,1,B) (0,0,C) (2,1,B) (5,3,EOF)`

Tendo esta sequência de triplas, pode ainda verificar-se qual a frequência com que surgem os vários números que compõem as triplas. Se a frequência não for uniforme, pode aplicar-se um segundo algoritmo de compressão (e.g., Huffman).

Secção 2

Algoritmo Lempel-Ziv (1978)

- É um código universal.
- É um código de dicionário.
- O output do algoritmo é um par `(W,S)` que consiste numa palavra de código (índice no dicionário) seguido de um símbolo.
- Em cada passo de compressão é acrescentada uma palavra nova ao dicionário, que é a extensão de uma já existente. A palavra nova é caracterizada pelo par `(W,S)`.
- A vantagem relativamente ao LZ77 é o número mais reduzido de comparações necessárias.

- O dicionário está inicialmente vazio.
- Lê uma sequência de símbolos até que essa sequência não esteja no dicionário.
- Adiciona essa sequência ao dicionário, atribuindo-lhe uma palavra de código nova (índice).
- Output = `(W,S)`, onde
  - `W` é o índice do prefixo da sequência no dicionário (o prefixo é a sequência menos o último símbolo).
  - `S` é o último símbolo da sequência, e que a torna não existente no dicionário.

Exemplo

Input: ABBCBCABA.

0	" "
1	A
2	B
3	BC
4	BCA
5	BA

Output: (0,A) (0,B) (2,C) (3,A) (2,A)

Secção 3

Algoritmo Lempel-Ziv-Welch (1984)

- É um melhoramento sobre o LZ78.
- O facto de ter sido patenteado reduziu a sua adoção. (Patente expirou em 2004)
- Diferenças relativamente ao LZ78:
  - O output é formado apenas por palavras de código.
  - O dicionário é inicializado com todo o alfabeto.
  - Prefixo da nova sequência é o último símbolo da sequência anterior.

LZW - Algoritmo de compressão

- Inicialmente:
  - Prefixo vazio `P=‘‘ ’’`
  - Dicionário contém todos os símbolos do alfabeto.
- Lê símbolos `S` enquanto `P+S` está no dicionário (`S` é o último símbolo).
- Output = código de `P`.
- Adiciona `P+S` ao dicionário.
- Próximo prefixo é o último símbolo, `P=S`.
- Repetir.

LZW - Exemplo de compressão

Exemplo

Input: AABABAABABAB

1	A
2	B

Output:

Exemplo

Input: AABABAABABAB

1	A
2	B
3	AA

Output: 1

Exemplo

Input: `AABABAABABAB`

1	A
2	B
3	AA
4	AB

Output: `1,1`

LZW - Exemplo de compressão

Exemplo

Input: AABABAABABAB

1	A
2	B
3	AA
4	AB
5	BA

Output: 1,1,2

LZW - Exemplo de compressão

Exemplo

Input: `AABABAABABAB`

1	A
2	B
3	AA
4	AB
5	BA
6	ABA

Output: `1,1,2,4`

LZW - Exemplo de compressão

Exemplo

Input: `AABABAABABAB`

1	A
2	B
3	AA
4	AB
5	BA
6	ABA
7	AAB

Output: `1,1,2,4,3`

LZW - Exemplo de compressão

Exemplo

Input: `AABABAABABAB`

1	A
2	B
3	AA
4	AB
5	BA
6	ABA
7	AAB
8	BAB

Output: `1,1,2,4,3,5`

LZW - Exemplo de compressão

Exemplo

Input: `AABABAABABAB`

1	A
2	B
3	AA
4	AB
5	BA
6	ABA
7	AAB
8	BAB

Output: `1,1,2,4,3,5,8`

LZW - Algoritmo de descompressão

- Inicialmente:
  - Dicionário contém todos os símbolos do alfabeto.
  - "Actual" é o primeiro número.
  - Descodifica e output.
- Lê número "Próximo",
- Descodifica e output,
- Adiciona ao dicionário "Actual" + 1º símbolo do "Próximo",
- "Actual" $\leftarrow$ "Próximo"
- Repetir.

LZW - Exemplo de descompressão

Exemplo

Input: 1124358

1	A
2	B

Output: `A`

Inicialização: "Actual"=1, output=dict[1]=A.

Exemplo

Input: 1124358

1	A
2	B
3	AA

Output: `A A`
"Actual"=1,
"Próximo"=1, output=dict[1]=A,
dict[3]= "Actual" + 1° símbolo de "Próximo"=AA

Exemplo

Input: 1124358

1	A
2	B
3	AA
4	AB

Output: `A A B`
"Actual"=1,
"Próximo"=2, output=dict[2]=B,
dict[4]= "Actual" + 1° símbolo de "Próximo"=AB

Exemplo

Input: 1124358

1	A
2	B
3	AA
4	AB
5	BA

Output: `A A B AB`
"Actual"=2,
"Próximo"=4, output=dict[4]=AB,
dict[5]= "Actual" + 1° símbolo de "Próximo"=BA

Exemplo

Input: 1124358

1	A
2	B
3	AA
4	AB
5	BA
6	ABA

Output: `A A B AB AA`
"Actual"=4,
"Próximo"=3, output=dict[3]=AA,
dict[6]= "Actual" + 1° símbolo de "Próximo"=ABA

Exemplo

Input: 1124358

1	A
2	B
3	AA
4	AB
5	BA
6	ABA
7	AAB

Output: `A A B AB AA BA`
"Actual"=3,
"Próximo"=5, output=dict[5]=BA,
dict[7]= "Actual" + 1° símbolo de "Próximo"=AAB

Exemplo

Input: 1124358

1	A
2	B
3	AA
4	AB
5	BA
6	ABA
7	AAB
8	???

Output: `A A B AB AA BA ???`
"Actual"=5,
"Próximo"=8, output=dict[8]=???,
dict[8]= "Actual" + 1° símbolo de "Próximo"=BA + ?
Como sabemos que "Próximo"=BA+?, então o primeiro símbolo é B e portanto dict[8]=BAB.

LZW - Exemplo de descompressão

Exemplo

Input: 1124358

1	A
2	B
3	AA
4	AB
5	BA
6	ABA
7	AAB
8	BAB

Output: `A A B AB AA BA BAB`
"Próximo"=8, output=dict[8]=BAB.

Exemplo

Input: 1124358

1	A
2	B
3	AA
4	AB
5	BA
6	ABA
7	AAB
8	BAB

Output: `A A B AB AA BA BAB`
"Próximo"=8, output=dict[8]=BAB.

Conclusão: se "Próximo" refere-se a uma linha ainda não existente no dicionário, ficamos a saber que o primeiro símbolo é igual ao primeiro símbolo de "Actual".