# An LZ-Index Oriented to Natural Language

Pedro Morales
M.Sc. Student, DCC
Advisor: Gonzalo Navarro

# Introduction

- Natural language: notion of word
- Efficient word indexes: inverted index
- Great development in full-text indexes
- Compressed self indexes

# Proposal

- Modify existing compressed self index solution
- Match space use
- Match search performance
- Make further improvements for natural language

# Inverted index

- Three main structures:
  - Vocabulary
  - List of occurrences
  - Text itself
- Small size

# Inverted index

This is an example text. For example we have words.

1 6 9 11 19 25 26 27 30 35

| Vocabulary | Occurrences |
| --- | --- |
| example | 11 26… |
| text | 19… |
| words | 35… |

# Inverted index: Search

- Search procedure:
  - Search each word in vocabulary
  - Obtain lists of occurrences
  - Merge of lists, if needed
- Report complexity for two word query:
  - $O(occ(w_1)\log(occ(w_2)))$

# Inverted index: Size

- Many different granularities:
  - Full inverted index
  - Block addressing index
- Trade-off between space and performance
- Average index size: 20% of text
- Text can be compressed

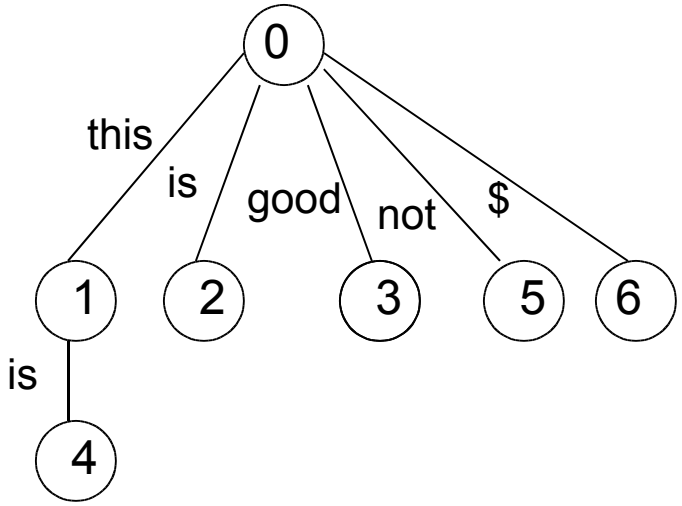
# LZ-Index

- Based in Ziv-Lempel compression
- Designed for fast text recovery
- 4 main structures:
  - LZTrie
  - RevTrie
  - Node
  - Rnode

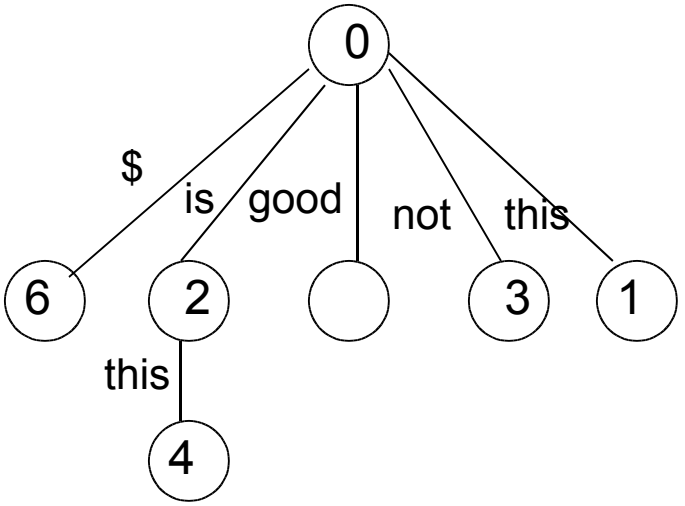
# LZ-Index: Modifications

- Word index
- Vocabulary becomes alphabet
- Non-Words (separators) are not indexed
- Separators must be kept

# LZ-Index




LZTrie

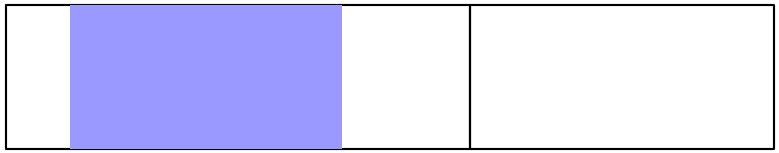



RevTrie

| This | is | good | This is | not | $ |
|---|---|---|---|---|---|
| 1 | 2 | 3 | 4 | 5 | 6 |

# LZ-Index: Search

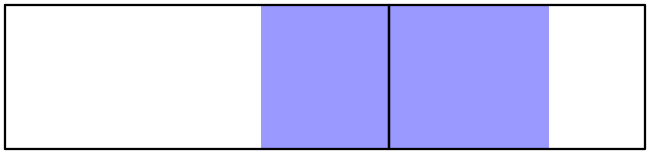
- Type 1 occurrences:



  - Search reverse pattern in RevTrie
  - Obtain blocks finishing with pattern
  - For each block report subtree in LZTrie

# LZ-Index: Search (2)

- Type 2 occurrences:



  - Split pattern
  - Search blocks starting and finishing with each part
  - Find pairs of successive blocks
  - Report pairs

# LZ-Index: Search (3)

- Report complexity for two word query:
  - Min(occ($w_1$), occ($w_2$))

# LZ-Index: Size Estimation

- Words:
  - Average word length: 7 chars
  - n/7 pairs word-separator
  - Estimated vocabulary $V = Kn^{\alpha}, K = 40, \alpha = 0.5$
  - Word based LZ compression: 20% of size of text
- Separators:
  - Stored using Huffman coding
  - Due to slant $Zipf \approx 1/i^2$
  - $H_0 = 2\sum \frac{1}{i^2} \log i$

# LZ-Index: Estimation (2)

- Estimated compression for separators: 15% of size of text
- Estimated size of LZ-Index 4 times entropy of text
- Total size of index and separators: 95% size of text

# Implementation difficulties

- Large alphabet requires algorithm modification
  - Method used to find child depends on arity
- Presentation problem:
  - Separator and stop words

# Further improvements

- Vocabulary modifiers
  - Capital letters
  - Accented letters
  - Stemming
- Keep word variants Huffman coded

Thank you.