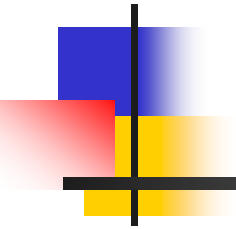
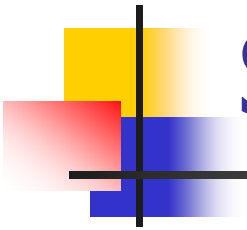
Compression of a Dictionary



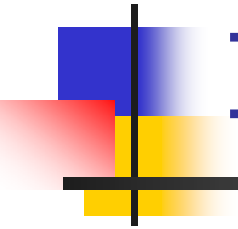
Jan Lánský, Michal Žemlička
zizelevak@matfyz.cz
michal.zemlicka@mff.cuni.cz

Dept. of Software Engineering
Faculty of Mathematics and Physics
Charles University



Synopsis

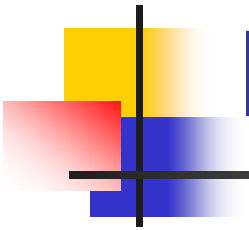
- Introduction
- Existing methods
- Trie-based methods
- Results
- Conclusion



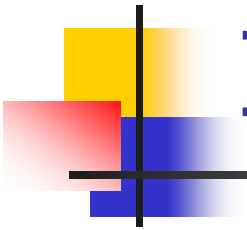
Introduction

Why we are compressing a dictionary ?

Large Alphabet Compression

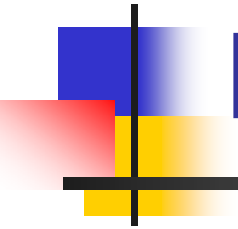


- Text Files - Compression over alphabet of words or syllables.
- Alphabet (Dictionary) must be transferred with the coded message
- Word-based methods
  - Moffat 1989
- Syllable-based methods
  - Lánský, Žemlička 2005



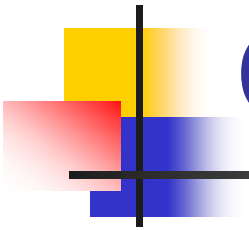
Influence of File Size

- Large files
  - Dictionary takes **small** part of message
  - Influence of compression of the dictionary on compression ratio is **small**
- Small files
  - Dictionary takes **large** part of message
  - Influence of compression of the dictionary on compression ratio is **large**



Existing methods

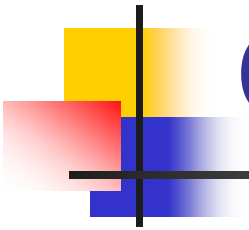
Common used methods for compression of a dictionary of words or syllables



Character by Character (CD)

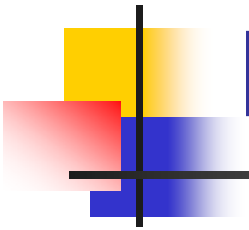
- Code of string is composed from
  - Code of string type
    - Moffat: 2 types of words (word, non-word)
    - Lánský, Žemlička: 5 types of syllables
  - Encoded length of the string
  - Symbol codes

Character by Character (CD)

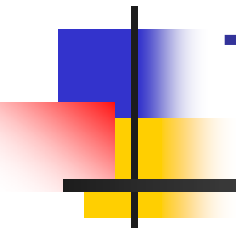


- Examples
  - code("**to**") = codeType(**lower**), codeLength(**2**), codeLower('**t**'), codeLower('**o**')
  - code("**153**") = codeType(**numeric**), codeLength(**3**), codeDigit('**1**'), codeDigit('**5**'), codeDigit('**3**')

External Compression

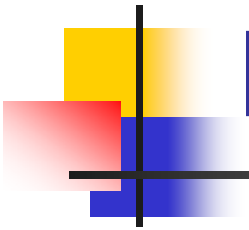


- All strings from dictionary are concatenated by using separator
- This resulting string is compressed by
  - LZW (we denote LZWD)
  - Bzip2 (we denote bzipD)
  - …



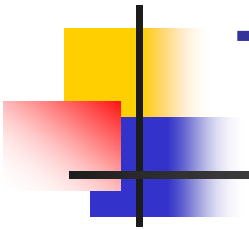
Trie-based methods TD1, TD2, TD3

Compression of a dictionary using its structure



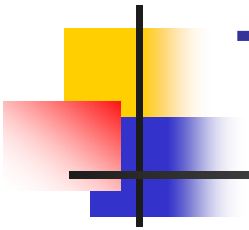
Dictionary

- Data structure trie
  - Nodes may represent strings
  - Father represents a prefix of its sons
- Mapping between strings and its **order** is unique in whole dictionary
- **Order** is obtained during compression



Trie data structure

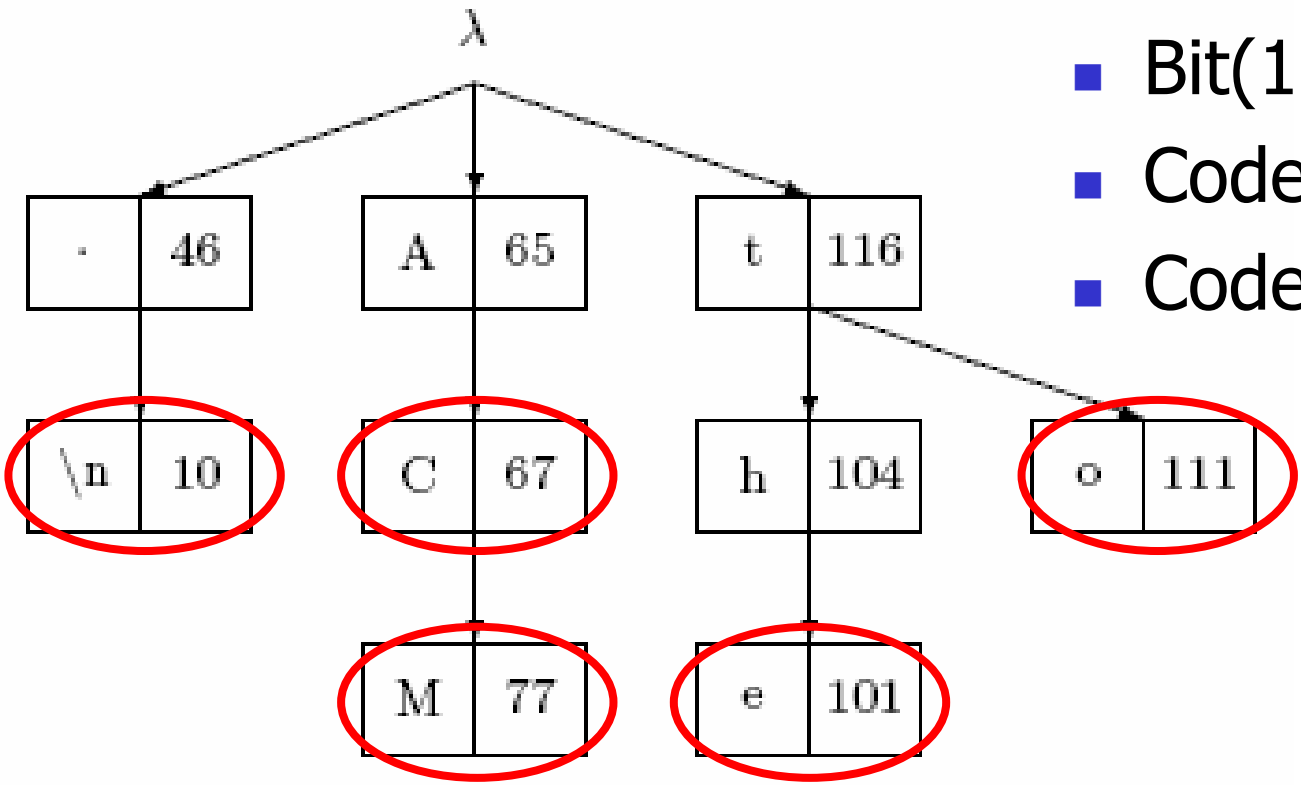
- For each node we know
  - Whether a node represents a string (**represents**)
  - Number of sons (**count**)
  - Array of sons (**son**)
  - Extension of each son (**extension**)



TD1 - encoding

- EncodeTD1 ()
  - EncodeGamma number of sons **count**
  - Encode **represents** (bit 0 or 1)
  - For each son **s**
    - Distance = **s**.extension – previous(**s**).extension
    - EncodeDelta(Distance)
    - EncodeNode(**s**)

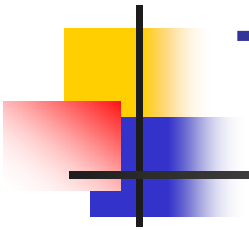
TD1 - Example



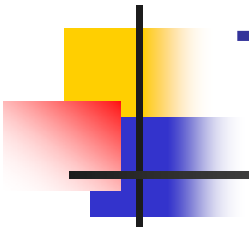
- ... Code node '**C**':
- Code(1) – count
- Bit(1) – repr.
- Code(67-0) – dist
- Code node '**M**' ...

- Dictionary: "the", "to", "ACM", "AC", ".\n "

TD2 - Improvement



- In TD1 version the distances between sons are coded.
  - Distances are calculated according binary values of the extending symbols
- These distances are encoded by Elias delta coding representing
  - smaller numbers by shorter codes
  - larger numbers by longer codes.
- Goal – decrease distances

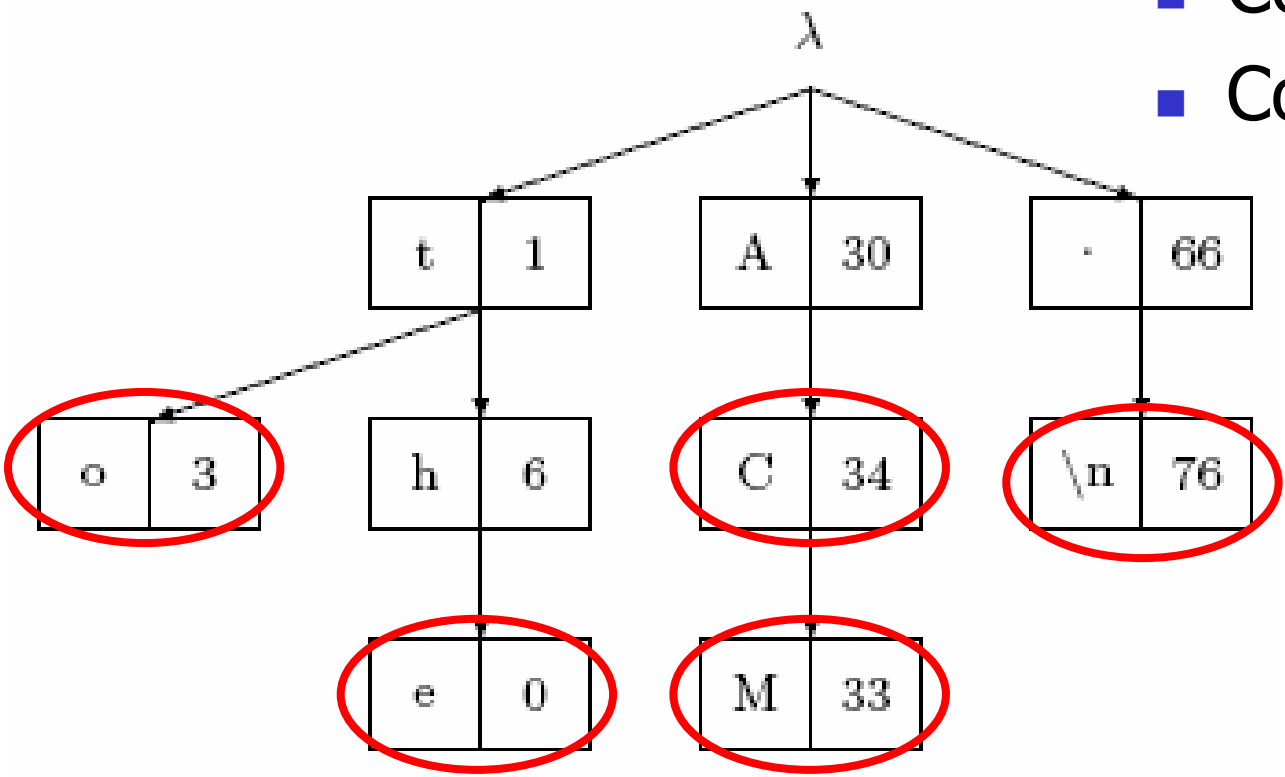


TD2 - Improvement

- Reordering alphabet
  - Primary according symbol type
  - Secondary according symbol frequency
  - 0-27 lower-case letter, 28-53 upper-case letters, 54-63 digits, 64-255 other symbols
- TD2 - Distances between sons are counting in this new alphabet
  - TD2 gives shorter distances and its codes

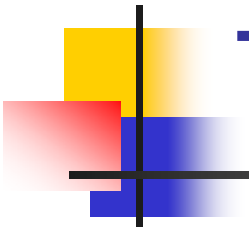
TD2 - Example

- ... Code node '**C**':
- Code(1) – count
- Bit(1) – repr.
- Code(34-0) – dist
- Code node '**M**' ...



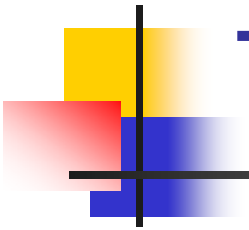
- Dictionary: "the", "to", "ACM", "AC", ".\n "

TD3 - Improvement



- 5 types of words and syllables
  - Lower ("hour")
  - Upper ("HOUR")
  - Mixed ("Hour")
  - Numeric ("123")
  - Other ("???")
- After coding 1-2 symbols from a string we can determine its type and improve its coding
  - 2 symbols per Mixed/ Upper, 1 symbol otherwise

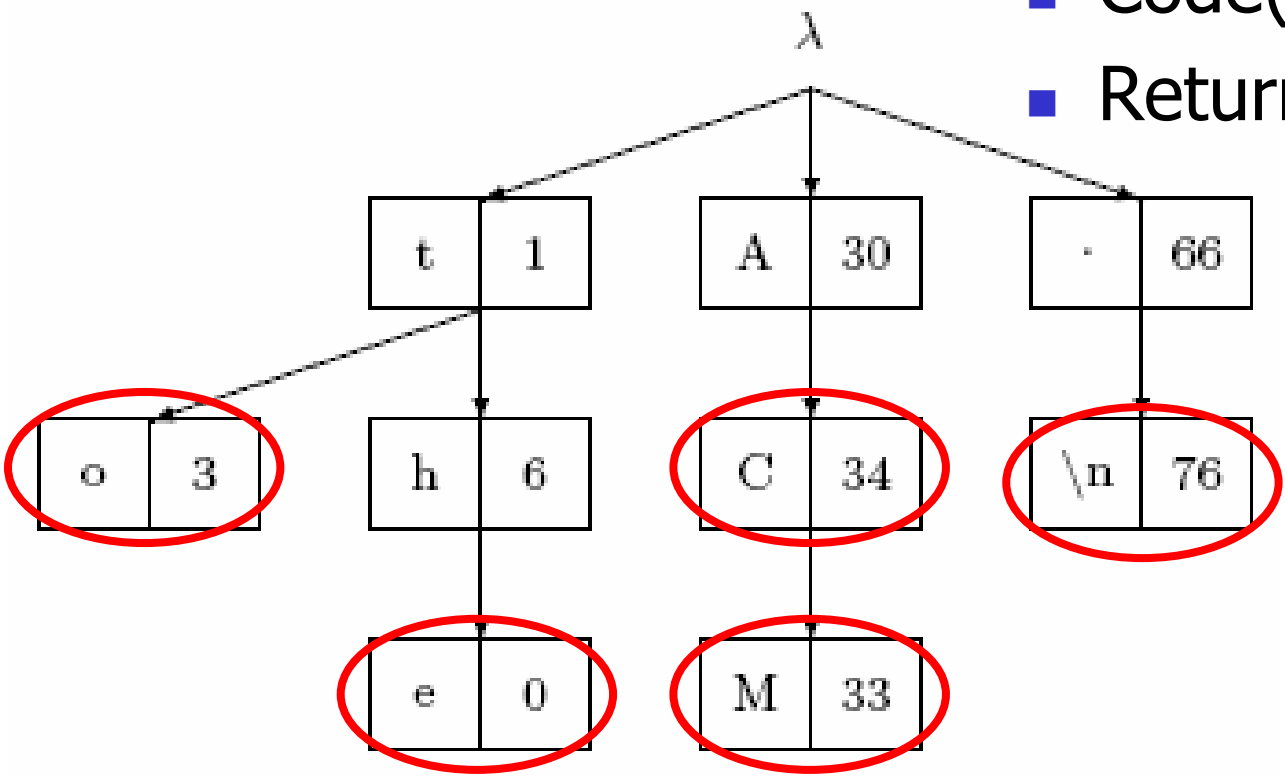
TD3 - Improvement



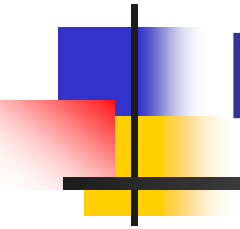
- Function **first**
  - First(lower-case letter) = 0
  - First(upper-case letter) = 28
  - First(digit) = 54
  - First(other) = 64
- TD3 – if we know the type of the string, we decrease the distance of the first son by the value of function **first** for the son extension

TD3 - Example

- ... Code node '**M**':
- Code(1) – count
- Bit(1) – repr.
- Code(33-28-0) – dist
- Return to node '**C**' ...



- Dictionary: "the", "to", "ACM", "AC", ".\n "

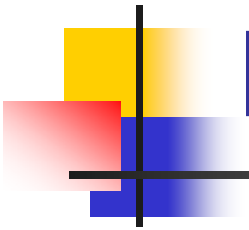


Results

Comparison of TD1, TD2, TD3, CD, LZWD and BzipD on dictionaries of words and syllables in Czech, English and German

Results - syllables

— / Lang.	File size / Method	100 B 1 kB	1 kB 10 kB	10 kB 50 kB	50 kB 200 kB	200 kB 500 kB	500 kB 2MB	2 MB 5 MB
CZ	LZWD	5.359	3.233	1.423	0.562	0.343	0.204	—
CZ	CD	3.741	2.432	1.130	0.461	0.284	0.169	—
CZ	BzipD	5.285	2.952	1.227	0.468	0.285	0.168	—
CZ	TD1	4.124	2.232	0.870	0.315	0.185	0.115	—
CZ	TD2	2.944	1.594	0.638	0.240	0.143	0.093	—
CZ	TD3	**2.801**	**1.532**	**0.612**	**0.226**	**0.134**	**0.081**	—
EN	LZWD	4.580	1.715	0.732	0.426	0.269	0.152	0.059
EN	CD	2.983	1.287	0.583	0.360	0.234	0.133	0.052
EN	BzipD	4.390	1.523	0.626	0.353	0.222	0.124	0.047
EN	TD1	3.792	1.276	0.506	0.272	0.158	0.086	0.033
EN	TD2	2.871	0.954	0.384	0.212	0.124	0.069	0.028
EN	TD3	**2.666**	**0.890**	**0.354**	**0.195**	**0.116**	**0.063**	**0.024**
GE	LZWD	4.259	2.995	1.139	0.580	0.345	0.202	0.104
GE	CD	3.068	2.360	0.997	0.530	0.315	0.185	0.091
GE	BzipD	4.127	2.689	0.949	0.479	0.285	0.166	0.087
GE	TD1	3.952	2.539	0.832	0.377	0.207	0.122	0.045
GE	TD2	3.020	1.914	0.627	0.284	0.157	0.097	0.035
GE	TD3	**2.730**	**1.805**	**0.599**	**0.275**	**0.150**	**0.086**	**0.033**



Results - syllables

- TD3 outperforms other methods on all languages and file sizes
  - Syllables are short
  - Trie of syllables is dense
- Example
  - 10Kb Czech file
  - 770 bytes of dictionary by TD3
  - 1540 bytes of dictionary by CD (second best)

Results - words

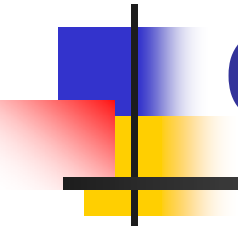
— / Lang.	File size / Method	100 B 1 kB	1 kB 10 kB	10 kB 50 kB	50 kB 200 kB	200 kB 500 kB	500 kB 2MB	2 MB 5 MB
CZ	LZWD	5.984	4.549	3.076	1.934	1.557	1.161	—
CZ	CD	**4.378**	**3.830**	2.948	1.968	1.648	1.260	—
CZ	BzipD	5.784	4.045	**2.559**	1.582	1.255	0.921	—
CZ	TD1	8.443	6.520	4.146	2.250	1.713	1.178	—
CZ	TD2	5.935	4.531	2.874	1.550	1.176	0.814	—
CZ	TD3	5.781	4.462	2.844	**1.534**	**1.167**	**0.800**	—
EN	LZWD	4.699	2.195	1.203	0.872	0.687	0.443	0.189
EN	CD	**3.100**	**1.776**	1.095	0.847	0.695	0.454	0.197
EN	BzipD	4.508	1.915	**1.002**	**0.714**	0.563	0.361	0.154
EN	TD1	6.320	3.144	1.698	1.108	0.813	0.498	0.191
EN	TD2	4.526	2.142	1.144	0.753	0.554	0.341	0.132
EN	TD3	4.219	2.062	1.110	0.734	**0.544**	**0.333**	**0.128**
GE	LZWD	4.712	3.634	1.819	1.227	0.996	0.706	0.716
GE	CD	**3.582**	**3.091**	1.787	1.293	1.096	0.799	0.789
GE	BzipD	4.409	3.216	**1.506**	**1.001**	**0.797**	**0.558**	0.565
GE	TD1	7.187	5.748	2.585	1.700	1.383	0.945	0.844
GE	TD2	4.985	3.885	1.691	1.094	0.875	0.601	0.534
GE	TD3	4.699	3.776	1.660	1.085	0.867	0.591	**0.532**

Results - words

- Czech
  - On 50kB and larger files is TD3 best
  - **Long** words, **dense** trie of words
- English
  - On 200kB and larger files is TD3 best
  - **Short** words, quite **dense** trie of words
- German
  - On 2MB and larger files is TD3 best
  - **Long** words, quite **sparse** trie of words

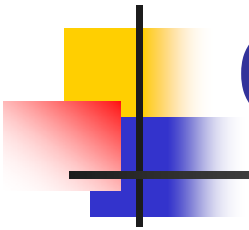
Results - words

- How are methods succesfull on?
- Smaller files
  - 1. CD, 2.-3.TD3, 2.-3. BzipD, 4. LZWD
- Middle-sized files
  - 1. BzipD, 2. TD3, 3. CD, 4. LZWD
- Larger files
  - 1. TD3, 2. BzipD, 3. CD, 4. LZWLD



Conclusion

On what types of dictionaries is TD3 good ?



Conclusion

- Where is TD3 successful
  - Dense tries with short string
  - Dictionaries of syllables
  - Larger dictionaries of words
- TD3 is not bad on other types of dictionaries
  - TD3 is usually at least the second best method