Towards Perfection of Machine Learning of Competing Patterns: The Use Case of Czechoslovak Patterns Development

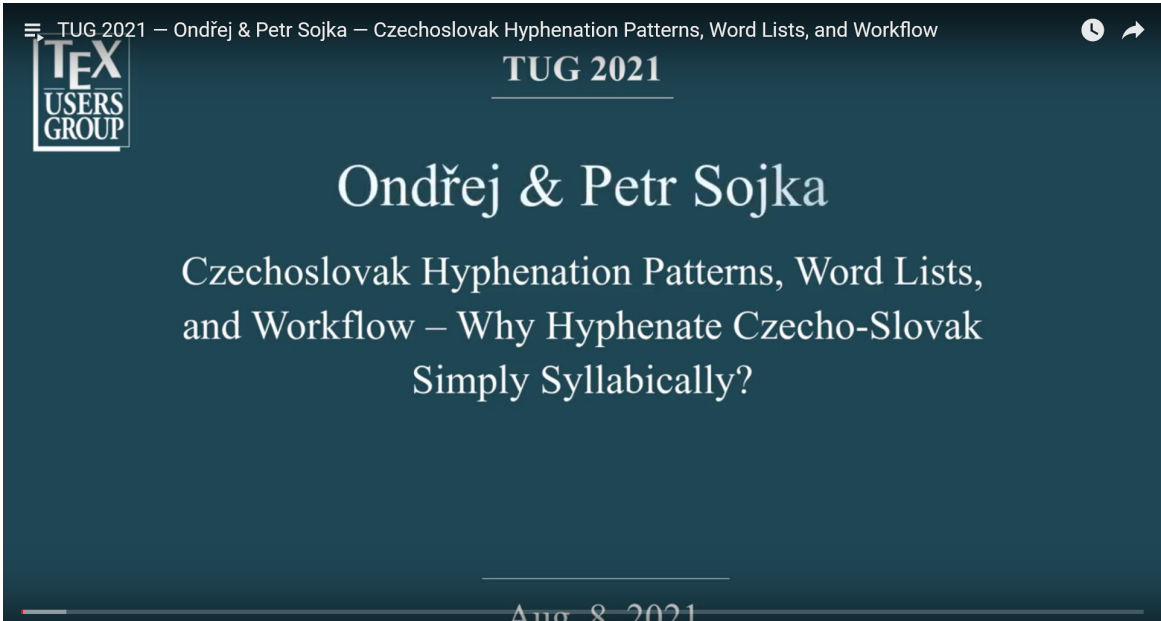
Ondřej Sojka, Petr Sojka

Faculty of Informatics, Masaryk University

December 9, 2023


EXIT

TUG 2021 – one set of patterns for 2 languages: Czech and Slovak



Contents

Section 1

Dictionary Problem Solution by Competing Patterns

Dictionary Problem

Finding space- and time-effective or even *optimal* solution to store the key-value pairs set.

word $\rightarrow$ word-related data

hy-phen-ation $\rightarrow$ 2 6

* * *

Solution 1: tr[ie]e-based solutions [1, 2]
Time: $O(1)$ for both tree-based solutions (constant C is the tree depth list) and
Space: $O(D)$, dictionary storage size D

Solution 2: Hashing
Same complexities, C relates to hash computation time with perfect hashing.
The absolute values of C and the linear coefficient for D are important.

Patterns (of hyphenation) that compete with each other

Frank Liang, DEK's student at Stanford (Ph.D., 1983), developed a general, language-independent method and algorithm for hyphenation based on the idea of competing patterns of varying length to cope with exceptions. [3].

- pattern is a substring with a piece of information about hyphenation between characters: `hy3ph he2n n2at hen5at`
- odd numbers permit, even numbers forbid hyphenation
- patterns are as short as possible to be as general as possible (new compound words, etc.)
- pattern compete with each other: instead of one big set of patterns, decomposition into layered sets generated in *levels*
  p_1 hyphenating patterns generated in level 1, p_2 inhibiting patterns—exceptions for p_1),
  p_3 hyphenating patterns to cover what has not been covered by "$p_1 \wedge \neg p_2$"),...

Hyphenation lookup: an instance of dictionary problem

```
    h y p h e n a t i o n
p1          1n a
p1              1t i o n
p2           n2a t
p2                2i o
p2       h e2n
p3  h y3p h
p4       h e n a4
p5       h e n5a t
    h0y3p0h0e2n5a4t2i0o0n
```

hy-phen-ation $\rightarrow$ 2 6
$\ldots \rightarrow \ldots$
$\ldots \rightarrow \ldots$
key $\rightarrow$ data

The solution to the dictionary problem:
For the key part (the word) to store
the data part (its hyphenation)

Given the already hyphenated word list of a language (dictionary), *how to generate the patterns*?
Liang's task was: less than 5,000 patterns, less than 30,000 bytes per language in format file (RAM during TEX run).

hyphen.tex generation by patgen (Liang, 1983) [3]

level	parameters	patterns	good	bad	good	bad
1	1 2 20 (4)	458	67,604	14,156	76.6%	16.0%
2	2 1 8 (4)	509	7,407	11,942	68.2%	2.5%
3	1 4 7 (5)	985	13,198	551	83.2%	3.1%
4	3 2 1 (6)	1647	1,010	2,730	82.0%	0.0%
5	1 ∞ 4 (8)	1320	6,428	0	**89.3%**	0.0%

A total of 4,919 patterns (4,447 unique) were obtained in hyphen.tex (27,860 bytes) from Webster's Pocket dictionary (30,000+ words only). *Suffix-compressed packed trie* occupying 5,943 locations, with 181 outputs. It occupies *less than 1%* of original word list, e.g. several orders of magnitude compression ratio.

Precision close to 100% (errors as exceptions published in TUGBoat regularly), and **Recall** is 89.3%, e.g. patterns do not find more than 10% of permissible hyphens.

And **generalization** abilities on top of that!

patgen program: machine learning from data

One of the very first approaches that harnessed the power of data: Liang's program patgen for generation of hyphenation patterns from a word list:

- efficient lossy or lossless *compression* of hyphenated dictionary with several orders of magnitude compression ratio.
- generated patterns have minimal length, e.g., shortest context possible, which results in their *generalization* properties.
- hyphenation of out-of-vocabulary words, too.

For Czech, *exact lossless* pattern generation *is feasible* [6] (TUG 2019), while reaching *100% coverage and simultaneously no errors*, and the same holds for 2 language patterns (Czech+Slovak). [5]
Strict pattern minimality (size) is not an issue nowadays but improves generalization.

Section 2

State of the Art

The “low-hanging” fruits in question

- ☒ no adoption of ’modern’ machine learning methods such as 10-fold cross-validation, data augmentation [7]
- ☒ old, small datasets with insufficient English representation
- ☒ are multilingual patterns possible?
- ☐ critical parameters picked according to vibes
- ☐ doesn’t the lack of hyphenation at the first and last location add complexity?
- ☐ unmaintained patterns for many languages

Grid search over parameters

Table 1: Parameters found by grid search on wordlist dataset 2023uniqlr1. Generalization metrics: Good: 99.58%, Bad: 0.86%, Missed: 0.42%, Precision: 0.9915, Recall: 0.9958, F-Score ($\beta = 1/7$): 0.9916, F-Score ($\beta = 1/77$): 0.9915

Level	Patterns	Good	Bad	Missed	Lengths	Params
1	1,889	1,625,346	276,301	41,897	1 3	1 31
2	1,872	1,620,864	4,504	46,379	1 3	1 41
3	3,886	1,667,204	5,414	39	2 6	1 31
4	729	1,666,973	0	270	2 7	1 41

Section 3

The Data Consistency Cleanup

Why?

- no hyphenations at first two and last two hyphenation points \lefthyphenmin=\righthyphenmin=2 → \lefthyphenmin=\righthyphenmin=1
- doesn't this confuse the patterns due to distorted semantic concept of syllables (.V-, V-V. patterns, CV, CCV, CCCV patterns)?
- problem: no hyphenations at first two and last two hyph. points

Problem: no hyphenation points marked after one vowel syllable at the beginning word (`o-me-le-ta é-te-ric-ký`) and before one vowel syllable at the end of the word.

Marking syllable hyphenation points at word border

```
grep ^[aeiouy][aáeéiíoóuúůyý] czechoslovak-lr1.wlh |
uniq |
grep -v ^..-|
grep -v ^[aáeéiíoóuúůyý][^aáeéiíoóuúůyý][^aáeéiíoóuúůyý] |
grep -v ^..$ |
grep -v ^[aáoó][úůu] |
grep -v ^[eé][ií]
```

10-fold cross-validation results

Parameters	Good	Bad	Missed	Patterns	Precision	$F_{1/7}$
custom2020	99.40%	0.75%	0.60%	5,124	0.9925	0.9925
correctopt2020	99.57%	0.83%	0.42%	8,384	0.9916	0.9917
sizeopt2020	99.11%	0.72%	0.88%	5,955	0.9927	0.9927
gridfound	99.60%	0.86%	0.40%	7,605	0.9914	0.9915

Section 4

Conclusion

Conclusion: RASLAN 2023 – finetuned, perfect version of Czechoslovak syllabification patterns

- ☒ no adoption of 'modern' machine learning methods such as 10-fold cross validation, data augmentation [7]
- ☒ old, small datasets with insufficient English representation
- ☒ are multilingual patterns possible?
- ☒ critical parameters picked according to vibes
- ☒ doesn't the lack of hyphenation at the first and last location add complexity?
- ☐ unmaintained patterns for many languages, caused by fragmentation of know-how

Section 5

Future Work

LONELY SCHOLAR™

CONVERSATIONS Issue 4

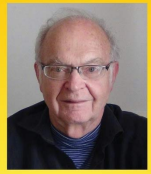
Algorithmic Barriers Falling

P=NP?

Donald E. Knuth

Edgar G. Daylight

Kurt De Grave (Ed.)



"A prize of $10 from Blum, $10 from Meyer, £4 from Paterson and 30-DM from Schnorr is offered to anyone who first solves the Cook-Karp problem whether P = NP.

Blum bet $100 that P ≠ NP against Paterson's $1 that P = NP."

— SIGACT News, January 1973, page 3

These playful bets capture youthful optimism in Complexity Theory.

Did Donald E. Knuth share this optimism at the time? And how did the findings of complexity theorists interlace with Knuth's pioneering work in the Analysis of Algorithms?

By investigating Knuth's developing thoughts on the theoretical underpinnings of efficient computation, this book sheds light on Knuth's present-day conjecture that P=NP.

LONELY SCHOLAR™
SCIENTIFIC BOOKS

ISBN 978-94-91386-04-6

9 789491 386046

Challenges

Minimal set cover problem reduction to dictionary problem. [4]
Word – covering all n members of set out of 2^n possible sets (e.g. words).
Pattern competition could be used for the *minimization* of set cover.

* * *

Development of Universal Syllabic Segmentation
CZ+SK -> CZ+SK+PL+RU+UA+... languages, possibly from the wordlist of Slava project, as bachelor project of Andrew

Section 6

Bibliography

Bibliography I

[1] Gianni Franceschini et al. “Implicit B-trees: a new data structure for the dictionary problem”. In: *Journal of Computer and System Sciences* 68.4 (2004). Special Issue on FOCS 2002, pp. 788–807. ISSN: 0022-0000. DOI: https://doi.org/10.1016/j.jcss.2003.11.003.

[2] Donald E. Knuth. *Sorting and Searching*. Third. Vol. 3. The Art of Computer Programming. Addison-Wesley, 1998.

[3] Franklin M. Liang. “Word Hy-phen-a-tion by Com-put-er”. PhD thesis. Stanford University, Aug. 1983, p. 44. URL: https://tug.org/docs/liang/liang-thesis.pdf.

[4] Petr Sojka. *From Minds to Pixels and Back (Habilitation Thesis)*. Masaryk University, Brno, Apr. 2008, pp. xvi+209.

Bibliography II

[5] Petr Sojka and Ondřej Sojka. “New Czechoslovak Hyphenation Patterns, Word Lists, and Workflow”. eng. In: *TUGboat* 42.2 (2021). ISSN: 0896-3207. URL: https://doi.org/10.47397/tb/42-2/tb131sojka-czech.

[6] Petr Sojka and Ondřej Sojka. “The Unreasonable Effectiveness of Pattern Generation”. In: *TUGboat* 40.2 (2019), pp. 187–193. URL: https://tug.org/TUGboat/tb40-2/tb125sojka-patgen.pdf.

[7] Petr Sojka and Ondřej Sojka. “Towards Universal Hyphenation Patterns”. In: *Proceedings of Recent Advances in Slavonic Natural Language Processing—RASLAN 2019*. Ed. by Aleš Horák, Pavel Rychlý, and Adam Rambousek. https://is.muni.cz/publication/1585259/?lang=en. Karlova Studánka, Czech Republic: Tribun EU, 2019, 63–68. URL: https://nlp.fi.muni.cz/raslan/2019/paper13-sojka.pdf.