New Czechoslovak Hyphenation Patterns, Word Lists, and Workflow

Why Hyphenate Czecho-Slovak?

Petr Sojka, Ondřej Sojka

Faculty of Informatics, Masaryk University

August 8, 2021
Contents

Introduction to Hyphenation Patterns

Czech, Slovak and Czechoslovak
  Universal Hyphenation Patterns?
  Hyphenation approaches

Pattern Generation

Conclusions

Bibliography
Section 1

Introduction to Hyphenation Patterns
Patterns (in general)

“pattern ORIGIN Middle English patron ‘something serving as a model’, from Old French. The change in sense is from the idea of patron giving an example to be copied. Metathesis in the second syllable occurred in the 16th cent. By 1700 patron ceased to be used of things, and the two forms became differentiated in sense.”


Patterns everywhere: rhythm patterns in music or poetry conveying message, patterns of behaviour, letter patterns, …, you name it: hyphenation patterns.
Patterns (of hyphenation) that compete each other

Frank Liang, DEK’s student at Stanford (Ph.D., 1983), developed the method and algorithms for hyphenation based on the idea of competing patterns of varying length to cope with exceptions. [3].

- general, language-independent method
- pattern is a substring with information about hyphenation between characters: hy3ph he2n n2at hen5at .euro7
- 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 each other: instead of one big set of patterns, decomposition into several layered generated sets: 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 “$p_1 \land \neg p_2$”,…
Hyphenation lookup: an instance of dictionary problem

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 TÉX run).
**Introduction to Hyphenation Patterns**

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

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

A total of 4,919 patterns (4,447 unique) obtained in hyphen.tex (27,860 bytes) from Webster pocket dictionary (30,000+ words only).  
*Suffix-compressed packed trie* occupying 5,943 locations, with 181 outputs (less than 1% of original word list).  
Patterns find 89.3% of the hyphens in the dictionary. 109 passes through the dictionary are needed.  
Generation required about 1 hour of CPU time on PDP-11.
Introduction to Hyphenation Patterns

**hyphen.tex used as a default for \language=0 in every \TeX installation**

\begin{verbatim}
% The Plain TeX hyphenation tables
% [NOT TO BE CHANGED IN ANY WAY!]
% Unlimited copying and redistribution of this file
% are permitted as long as this file is not modified.
% Modifications are permitted, but only if
% the resulting file is not named hyphen.tex.
patterns{% just type <return> if you’re not using INITEX
  .ach4 .ad4der .af1t .al3t .am5at
...
\end{verbatim}
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.

Generally, exact lossless pattern minimization is non-polynomial by reduction to the minimum set cover problem [5]. For Czech, exact lossless pattern generation is feasible [7] (TUG 2019), while reaching 100% coverage and simultaneously no errors.

Strict pattern minimality (size) is not an issue nowadays.
"An important feature of a learning machine is that its teacher will often be very largely ignorant of quite what is going on inside, although he may still be able to some extent to predict his pupil’s behaviour.” — Alan Turing, *Mind* 59:433–460, 1950
Section 2

Czech, Slovak and Czechoslovak
Why hyphenate Czecho-Slovak?

- separate, but very similar languages
- very practical
- 40+% of students at Faculty of Informatics, MU in Brno, Czech Republic, are Slovak

proof of concept for universal hyphenation patterns presented at RASLAN 2019 workshop [8]
Is there no word that has different hyphenations in the covered languages?

Then we can cover multiple languages with one set of hyphenation patterns.
Two approaches to hyphenation

- etymology-based
- phonology-based
Foreign words

- hyphenation must not disrupt reading
- English pronunciation: goo-gle
- pronunciation: go-o-g-le
Hyphenation in Czech

Rules are published at [2]:
https://prirucka.ujc.cas.cz/?id=135

- primarily syllabic according to pronunciation
- morphology only secondary (compound words)
- language per se, its vocabulary and hyphenation rules develop in time:
  roz-um (Haller, 1956, prefix roz, stem um) →
  ro-zum (2021, just syllables, etymology forgotten)\(^1\)

---

\(^1\) Similar shift is in other languages and cultures (UK→US)
Hyphenation in Slovak


- morphology primary according to the Ľ. Štúr Institute of Linguistics
- syllabic hyphenation secondary
- morphological boundaries are often also syllabic boundaries
- current patterns hyphenate mostly syllabically
- pupils learn to hyphenate syllabically
Observations

- telling apart the prefix *nej*- from *ne*- is a problem
- Slovak patterns hyphenate *syllabically* most of the time, contrary to recommendations of the Slovak language institute.
  - ne-na_u-čí
- is this correct behavior?
  - vy-ma-lo-va-ných
  - vy-maľ-o-va-ných
Tradeoffs

- impossible to hyphenate both languages “by the book”
- our patterns hyphenate syllabically
  - allows for easier reading
  - easier to hyphenate
  - lazier approach wins with language users
Section 3

Pattern Generation
We have

- good Czech patterns
- mediocre Slovak patterns (hand made, not generated) [1]
- Czech and Slovak word lists crawled and edited [8]
- Czech word list hyphenated with Czech patterns
- human hyphenators
We want a workflow, leading to

Patterns that *miss* very few or *no* hyphenation points and make *no* mistakes in either language
cssk-all-join.wls
(1,319 k CS+SK words)
patgen
(as hyphenator)
with cshyphen patterns
cssk-all-join.wlh
(1,319 k CS+SK hyphenated words)
cssk-all-intersect.wlh
(139 k words that are both in CS and SK)
diff
(fixing badly hyphenated SK words)
sk-corrections.wlh
(corrected SK words from cssk-all-intersect.wlh)
word lists union
(with added priorities)
(join 1x, intersect 2x, corrections 3x)
cssk-all-weighted.wlh
(1,319 k hyphenated words with weights)
csskhyphen.pat
patgen
(as pattern generator)
csskhyphen.pat
cs-sojka-correctopt.par
or cs-sojka-sizeopt.par
patgen
(as hyphenator)
with skhyph patterns
cssk-all-intersect.wls
(139 k words that are both in CS and SK)
cssk-all-intersect.wlh
(139 k hyphenated words that are both in CS and SK)
cssk-all-intersect.wlh
(139 k words hyphenated by Slovak patterns)
word lists union
(with added priorities)
(join 1x, intersect 2x, corrections 3x)
cssk-all-join.wls
(1,319 k CS+SK words)

patgen
(as hyphenator)
with cshyphen patterns

cssk-all-intersect.wls
(139 k words that are both in CS and SK)

patgen
(as hyphenator)
with skhyphen patterns

cssk-all-intersect.wlh
(139 k hyphenated words that are both in CS and SK)

cssk-all-join.wlh
(1,319 k CS+SK hyphenated words)

cssk-all-join.wls
(1,319 k CS+SK words)

patgen
(as hyphenator)
with cshyphen patterns

cssk-all-intersect.wlh
(139 k words that are both in CS and SK)

diff
(fixing badly hyphenated SK words)

sk-corrections.wlh
(corrected SK words from cssk-all-intersect.wlh)

word lists union
(with added priorities)
(join 1x, intersect 2x, corrections 3x)
cssk-all-join.wls
(1,319 k CS+SK words)
patgen
(as hyphenator)
with cshyphen patterns

cssk-all-join.wlh
(1,319 k CS+SK hyphenated words)
diff
(fixing badly hyphenated SK words)

word lists union
(with added priorities)
(join 1x, intersect 2x, corrections 3x)

cssk-all-weighted.wlh
(1,319 k hyphenated words with weights)
sk-corrections.wlh
(corrected SK words from cssk-all-intersect.wlh)
cssk-all-intersect.wlh
(139 k hyphenated words that are both in CS and SK)
cssk-all-intersect.wlh
(139 k words that are both in CS and SK)
diff
(fixing badly hyphenated SK words)

patgen
(as hyphenator)
with skhyph patterns
diff
(fixing badly hyphenated SK words)

sk-corrections.wlh
(corrected SK words from cssk-all-intersect.wlh)

patgen
(as pattern generator)
csskhyphen.pat
csskhyphen.pat

or

cs-sojka-correctopt.par
or cs-sojka-sizeopt.par
## Pattern generation – custom parameters

<table>
<thead>
<tr>
<th>Level</th>
<th>Patterns</th>
<th>Good</th>
<th>Bad</th>
<th>Missed</th>
<th>Lengths</th>
<th>Params</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>830</td>
<td>2,819,833</td>
<td>470,649</td>
<td>35,908</td>
<td>1 3</td>
<td>1 3 12</td>
</tr>
<tr>
<td>2</td>
<td>1,590</td>
<td>2,748,581</td>
<td>3,207</td>
<td>107,160</td>
<td>2 4</td>
<td>1 1 5</td>
</tr>
<tr>
<td>3</td>
<td>2,766</td>
<td>2,852,334</td>
<td>12,197</td>
<td>3,407</td>
<td>3 6</td>
<td>1 2 4</td>
</tr>
<tr>
<td>4</td>
<td>1,285</td>
<td>2,851,931</td>
<td>986</td>
<td>3,810</td>
<td>3 7</td>
<td>1 4 2</td>
</tr>
</tbody>
</table>

Custom parameters are used to filter out still manageable amount of words to check and fix in the primary word lists.
### Pattern generation – correct optimized parameters

<table>
<thead>
<tr>
<th>Level</th>
<th>Patterns</th>
<th>Good</th>
<th>Bad</th>
<th>Missed</th>
<th>Lengths</th>
<th>Params</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2,032</td>
<td>2,800,136</td>
<td>242,962</td>
<td>55,605</td>
<td>1 3</td>
<td>1 5 1</td>
</tr>
<tr>
<td>2</td>
<td>2,009</td>
<td>2,791,326</td>
<td>10,343</td>
<td>64,415</td>
<td>1 3</td>
<td>1 5 1</td>
</tr>
<tr>
<td>3</td>
<td>3,704</td>
<td>2,855,554</td>
<td>11,970</td>
<td>187</td>
<td>2 6</td>
<td>1 3 1</td>
</tr>
<tr>
<td>4</td>
<td>1,206</td>
<td>2,854,794</td>
<td>33</td>
<td>947</td>
<td>2 7</td>
<td>1 3 1</td>
</tr>
</tbody>
</table>

With correct optimized setting we get no errors (33 words could be added as patterns), with only 4 levels used.
10-fold cross validation results

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Good</th>
<th>Bad</th>
<th>Missed</th>
</tr>
</thead>
<tbody>
<tr>
<td>correctopt</td>
<td>99.81%</td>
<td>0.15%</td>
<td>0.04%</td>
</tr>
<tr>
<td>custom</td>
<td>99.64%</td>
<td>0.22%</td>
<td>0.14%</td>
</tr>
<tr>
<td>sizeopt</td>
<td>99.41%</td>
<td>0.18%</td>
<td>0.40%</td>
</tr>
</tbody>
</table>

Generalization abilities are checked by 10-fold cross validation: patterns are generated from 9/10 of word list, and performance is measured on yet unseen "new" words. This is repeated 10 times and the performance averaged.
# Pattern Generation

## Comparison – training (not validation) results

<table>
<thead>
<tr>
<th>Word list</th>
<th>Parameters</th>
<th>Good</th>
<th>Bad</th>
<th>Missed</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Slovak</td>
<td>[1, by hand]</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>20 kB</td>
</tr>
<tr>
<td>Czech</td>
<td>correctopt [7]</td>
<td>99.76%</td>
<td>2.94%</td>
<td>0.24%</td>
<td>30 kB</td>
</tr>
<tr>
<td>Czech</td>
<td>sizeopt [7]</td>
<td>98.95%</td>
<td>2.80%</td>
<td>1.05%</td>
<td>19 kB</td>
</tr>
<tr>
<td>Slovak</td>
<td>[6, Table 1]</td>
<td>99.94%</td>
<td>0.01%</td>
<td>0.06%</td>
<td>56 kB</td>
</tr>
<tr>
<td>Czechoslovak</td>
<td>sizeopt</td>
<td>99.67%</td>
<td>0.00%</td>
<td>0.33%</td>
<td>40 kB</td>
</tr>
<tr>
<td>Czechoslovak</td>
<td>correctopt</td>
<td>99.99%</td>
<td>0.00%</td>
<td>0.01%</td>
<td>45 kB</td>
</tr>
<tr>
<td>Czechoslovak</td>
<td>custom</td>
<td>99.87%</td>
<td>0.03%</td>
<td>0.13%</td>
<td>32 kB</td>
</tr>
</tbody>
</table>
Section 4

Conclusions
Summary of outcomes

1. new Czechoslovak patterns
2. new Czech and Slovak word lists
3. new workflow for hyphenated word list acquisition
4. new workflow for pattern development and customized pattern generation

All primary sources and generated patterns file csskhyphen.pat are posted in public github repository https://github.com/tensojka/cshyphen. Documentation is to be found in the paper in the TUG 2021 proceedings.
Next steps

- inclusion to tex-hyph package
- two possible approaches:
  - to introduce new ‘Czecho-Slovak language’ in babel, polyglossia, or
  - new patterns to be used just as replacements for both old Czech and Slovak patterns?
That’s it, folks! Comments, suggestions, pull requests are welcome!

Questions?
Section 5

Bibliography
Bibliography I


Bibliography
