New Czechoslovak Hyphenation Patterns, Word Lists, and Workflow*

Petr Sojka
Faculty of Informatics, Masaryk University, Brno, Czech Republic
sojka@fi.muni.cz

Ondřej Sojka
Faculty of Informatics, Masaryk University, Brno, Czech Republic
454904@mail.muni.cz

Abstract

Space- and time-effective segmentation and hyphenation of natural languages stay at the core of every document preparation system, web browser, or mobile rendering system. Recently, the unreasonable effectiveness of pattern generation has been shown – it is possible to use hyphenation patterns to solve the dictionary problem for a single language without compromise. In this article, we show how we applied the marvelous effectiveness of patgen for the generation of the new Czechoslovak hyphenation patterns that cover both Czech and Slovak languages. We show that the development of more universal hyphenation patterns is feasible, allowing for significant quality improvements and space savings. We evaluate the new approach and the new Czechoslovak hyphenation patterns for coverage and generalization.

“Any respectable word processing package includes a hyphenation facility. Those based on an algorithm, also called logic systems, often break words incorrectly.” Major Keary in [8]

1 Introduction

Hyphenation is at the core of every document preparation system, TeX or any modern web browser. Marchand et al. [11] showed that data-driven approaches to syllabification algorithms outperform rule-based ones, reaching accuracy around 95% per single language. Bartlett et al. [3] developed a machine learning approach for automatic syllabification, motivated by the needs of letter-to-phoneme conversion. Trogkanis et al. [24] used conditional random fields for word hyphenation and compared the accuracy and other metrics with the original technique of Liang [10]. Their results abstracted heuristics to optimize generated patterns by patgen, diminishing achievable performance by Liang’s technique.

Unicode Consortium supports 5,000 languages that are still in use today. A digital typographic system that supports Unicode and its languages in full should support hyphenation in the form of algorithms, rules, or patterns. Recently, there were attempts to tackle the word segmentation problem in different languages by Shao et al. [13]. The algorithm is error-prone, but it was developed primarily for speech recognition and language representation tasks. Due to the nonzero error rate, its applicability to the hyphenation task is limited. In a typesetting system, the hyphenation algorithm must cover all exceptions and not tolerate any errors.

Current hyphenation support based on hyphenation patterns is collected in the hyph-utf8 [12] project. The project uses ISO standards like Unicode and ISO 639-2. It contains hyphenation patterns for 65 different languages with an additional 9 dialect or transliteration variants. We do not know pattern performance — accuracy and coverage — for most patterns as there are no available word lists to use for the evaluation. These patterns have to be either loaded all in precomputed, compact form into TeX’s memory from format file at the start of every document compilation, which down its start-up time. Only LuaTeX allows loading patterns during runtime only for languages actually used in a document.

These patterns are supported in many programming languages, which makes their use straightforward.

Using these patterns in many programming languages (JavaScript, Perl, Python, C,….) is straightforward and makes the pattern usage rather versatile.

There are essentially two quite different approaches to hyphenation:

* This is updated and enriched version of paper published in the Zpravodaj ČSTUG [22].
etymology-based The rule is to cut a word on the border of a compound word or the border of the stem and an affix, prefix, or negation. A typical example are the British hyphenation rules by the Oxford University Press [1].


This paper evaluates the feasibility of the development of universal phonology-based (syllabic) hyphenation patterns. As a case study, we describe the development of Czechoslovak hyphenation patterns from word lists of Czech [15, 23, 16] and Slovak [18]. New patterns are generated from real world word lists and are superior to both Czech and Slovak patterns. We document our reproducible workflow and all resources in a public repository. We conclude by outlining further possible hyphenation pattern developments to meet demands of today.

“Hyphenation does not lend itself to any set of unequivocal rules. Indeed, the many exceptions and disagreements suggest it is all something dreamed up at an anarchists’ convention.” Major Keary in [8]

2 Methods
The core idea is to develop common hyphenation patterns for phonology-based languages. If these languages share pronunciation rules, homographs from different languages typically do not cause problems, as they are hyphenated the same. [6, 25] There are sporadic cases where the seam of a compound word dictates hyphenation point contrary to phonology (roz-um vs. ro-zum). These could be solved by not allowing the hyphenation of this particular word around this specific seam.

We recently showed that the approach to generate hyphenation patterns from word list by program patgen is unreasonably effective [21]. One can set the parameters of the generation process so that the patterns cover 100% of hyphenation points, and their size remains reasonably tiny. We compressed the word list with 3,000,000 hyphenated words into 30,000 bytes of the packed trie data structure for the Czech language. That means achieving a compression ratio of several orders of magnitude with 100% coverage and nearly zero errors [21]. For a similar language such as Slovak, the pronunciation is very similar, syllable-forming principles are the same, and compositional rules and prefixes are pretty close, if not identical.

We have decided to verify the approach by developing hyphenation patterns that will hyphenate both Czech and Slovak words without errors, with only a few missed hyphens. The missed hyphen will appear only in words like o-blit where meaning of the term is needed for the decision: o-blit or ob-lit. To resolve such ambiguities, the context-dependent decision will have to be done during runtime.

For the generation of these hyphenation patterns, we needed to create lists of correctly hyphenated Czech and Slovak words.

3 Data Preparation
For our work, Lexical Computing CZ donated word lists with frequencies for Czech and Slovak from the TenTen family of corpora [7, 9].

The Czech word list was cleaned up and extended as described by us [21, 19, 20], using the Czech morphological analyzer majka. In further processing, we used only words that appeared more than ten times. The final word list cs-all-cstenten.wls contained 606,494 words.

For Slovak, we obtained 1,048,860 Slovak words with a frequency higher than ten from 2011 SkTenTen corpora [7]. We only used words with a frequency higher than thirty that comprised only of ISO Latin 2 characters, obtaining file sktenten.wls with 544,609 words.

By joining both language files, we got 967,058 Czech and Slovak words in cssk-all-join.wls, of which 106,016 were contained in the intersection of both word lists: cssk-all-intersect.wls.

4 Pattern Development
Figure 1 illustrates the workflow of the Czechoslovak pattern development. We have used recent, accurate Czech patterns [21] for the hyphenation of the joint Czech and Slovak word list. We had to manually fix incorrect hyphenation, typically near the prefix and stem of words when phoneme-based hyphenation point was one character away from the seam of the prefix or compound word: neja-traktivnější, neja-teističtější, neje-kologičtější.

We have then hyphenated words used in both languages also by current Slovak patterns. There were only a few word hyphenations that needed to be corrected — we created the file sk-corrections.wlh that contained the fixed hyphenated words. Finally, we used them as an input to patgen with a higher weight during the generation of the final Czechoslovak hyphenated patterns.

We did not pursue 100% coverage at all costs because the source data is noisy, and we do not want the patterns to learn all the typos and inconsistencies. We expand on this in the Jupyter notebook [14]. Gentle readers may also find the scripts used there.
**Figure 1**: The development process of the new Czechoslovak patterns: Bootstrapping with Czech patterns, checking and fixing with a higher weight Slovak words that are common with Czech ones.
Table 1: Statistics from the generation of Czechoslovak hyphenation patterns with 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 1 3 12</td>
<td></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 1 1 5</td>
<td></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 1 2 4</td>
<td></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 1 4 2</td>
<td></td>
</tr>
</tbody>
</table>

Table 2: Statistics from the generation of Czechoslovak hyphenation patterns with 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 1 5 1</td>
<td></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 1 5 1</td>
<td></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 1 3 1</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>1,206</td>
<td>2,854,794</td>
<td>33</td>
<td>947</td>
<td>2 7 1 3 1</td>
<td></td>
</tr>
</tbody>
</table>

Table 3: Comparison of the efficiency of different approaches to hyphenating Czech and Slovak. Note that the Czechoslovak patterns are comparable in size and quality to single-language ones—there is only a negligible difference compared to i.e., purely Czech patterns.

<table>
<thead>
<tr>
<th>Word list</th>
<th>Parameters</th>
<th>Good</th>
<th>Bad</th>
<th>Missed</th>
<th>Size</th>
<th>Patterns</th>
</tr>
</thead>
<tbody>
<tr>
<td>Slovak</td>
<td>[4, by hand]</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>20 kB</td>
<td>2,467</td>
</tr>
<tr>
<td>Czech</td>
<td>correctopt [21]</td>
<td>99.76%</td>
<td>2.94%</td>
<td>0.24%</td>
<td>30 kB</td>
<td>5,593</td>
</tr>
<tr>
<td>Czech</td>
<td>sizeopt [21]</td>
<td>98.95%</td>
<td>2.80%</td>
<td>1.05%</td>
<td>19 kB</td>
<td>3,816</td>
</tr>
<tr>
<td>Slovak</td>
<td>[17, Table 1]</td>
<td>99.94%</td>
<td>0.01%</td>
<td>0.06%</td>
<td>56 kB</td>
<td>2,347</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>
<td>7,417</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>
<td>8,231</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>
<td>5,907</td>
</tr>
</tbody>
</table>

Table 4: Results of 10-fold cross-validation with evaluated parameters shows very good generalization properties (learning on 90%, and testing on remaining 10%).

<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>

5 Evaluation

We evaluated the quality of developed patterns by two metrics. Coverage of hyphenation points in the training word list tells how the patterns correctly predicted hyphenation points used in training. Generalization means how the patterns behave on unseen data, on the words not available in the data used during patgen training.

We see the coverage and generalization as a classification task, i.e., how the patterns classify hyphenation points in the training and testing word lists, respectively.

5.1 Classification

For evaluation of classification, there are four numbers in the contingency matrix that compare hyphenation point prediction by patterns with the ground truth expressed in the wordlist: true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN). In tables 1–3, we report:

Good sum or percentage of found hyphenation points (TP),

Bad sum or percentage of badly suggested hyphenation points (FP, type 1 error),

Missed sum or percentage of missed hyphenation points (FN, type 2 error).
Type 1 errors are clearly more severe than type 2 errors in our hyphenation points setup. Nonzero bad results do not necessarily mean that the patterns performed badly. The opposite is often the case—patterns have found a rule that is not obeyed in the ground truth wordlist. In other words, they found an inconsistency that needs to be fixed in the underlying word list rather than a valid exception. We practiced manual inspection of bad hyphenation points during the development of the word list.

5.2 Generalization

To assess the generalization properties, we used tenfold cross-validation, leaving one-tenth out of the training set to evaluate the patterns’ effectiveness on unseen words. We show results in Table 4. The evaluation metrics slightly differ with different patgen parameters, with the best results achieved when we maximize the coverage of the training set.

The achieved results show that both evaluation metrics are close to perfection. We can either opt for perfect coverage and reach it or push to maximize generalization qualities and performance on unseen words. In the first case, we essentially do lossless compression of wordlist hyphenation points by the developed pattern. In the second, we miss only less than 1% of valid hyphenation points. Achieving that for two languages in parallel seems like a good result. It is feasible to continue merging additional word lists to develop generic patterns for syllabically hyphenated languages.

“Esoteric Nonsense? Hyphenation is neither anarchy nor the sole province of pedants and pedagogues….Used in moderation, it can make a printed page more visually pleasing. If used indiscriminately, it can have the opposite effect, either putting the reader off or causing unnecessary distraction. If the intended audience is bound to read the work (a user manual, for example), poor hyphenation practice may not matter. If the author wants to attract and hold an audience, then hyphenation needs just as careful attention as any other aspect of presentation.” Major Keary in [8]

6 Conclusion and Future Works

We have shown that the development of common hyphenation patterns for several languages with similar pronunciations is feasible. Patgen was able to generalize hyphenation rules for both languages with a negligible increase in the size of the generated patterns.

The resulting Czechoslovak patterns hyphenate Czech and Slovak much better than the former single-language patterns, with higher coverage and zero error rate. The whole process is reproducible, is documented, and available as a Jupyter demo notebook with source code [14].

We offer the new patterns for “the Czechoslovak language” to the hyph-utf8 repository [12] and TEX Live distribution. One might further develop common patterns for additional syllabically hyphenated languages to be shared. Then it will be possible to hyphenate text for most of these languages without knowing what language they are written in. Most typesetting systems and browsers, including OpenOffice and Chrome, would use hyphenation in narrow columns. Most of them, if not all systems, use pattern technology and practices from the TEX community.

Acknowledgement

We are indebted to Don Knuth for questioning the common properties of Czech and Slovak hyphenation during our presentation of [21] at TUG 2019, which has led us in this research direction. We also thank everyone who commented on our workflow, patterns, and this paper.

References


