New Czechoslovak Hyphenation Patterns, Word Lists and Workflow*

Petr Sojka
Faculty of Informatics, Masaryk University, Brno, Czech Republic
sojka@fi.muni.cz
https://www.fi.muni.cz/usr/sojka/

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 [7]

1 Introduction

Hyphenation is at the core of every document preparation system, TEX or any modern web browser. Marchand et al. [10] 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. [22] used conditional random fields for word hyphenation and compared the accuracy and other metrics with the original technique of Liang [9]. 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 algorithms, rules, or language hyphenation patterns. Recently, there were attempts to tackle the word segmentation problem in different languages by Shao et al. [12]. The algorithm is error-prone, but it was developed primarily for speech recognition and language representation tasks. Due to the nonzero error rate, applicability to the hyphenation task is limited. In a typesetting system, the hyphenation algorithm must cover all exceptions and not tolerate the errors.

Current hyphenation support based on hyphenation patterns is collected in the `hyph-utf8` [11] 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 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 compilation, which slows down the start-up time of the document compilation. Only LuaTEX allows loading patterns during run-time only for languages actually used in a document. 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 CSTUG [20].

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 is the British hyphenation rules by the Oxford University Press [1].

phonology-based Hyphenation respects the pronunciation of syllables. It allows reading text with hyphenated lines almost as if the hyphenation were not there. American publishers [5] and the Chicago Manual of Style [2] users prefer this pragmatic approach.

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 [14, 21, 15] and Slovak [17]. New patterns are generated from actual 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 [7]

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. 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 [19]. 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 [19]. 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 `oblít` where *meaning* of the term is needed for the decision: `o-blít` or `ob-lít`. For such ambiguity resolution, the context-dependent decision will have to be done during runtime.

To generate 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 [6, 8].

The Czech word list was cleaned up and extended as described by us [19, 18], using the Czech morphological analyzer `majka`. We used only words that occurred more than ten times in further processing. 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 [6]. 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 [19] 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.

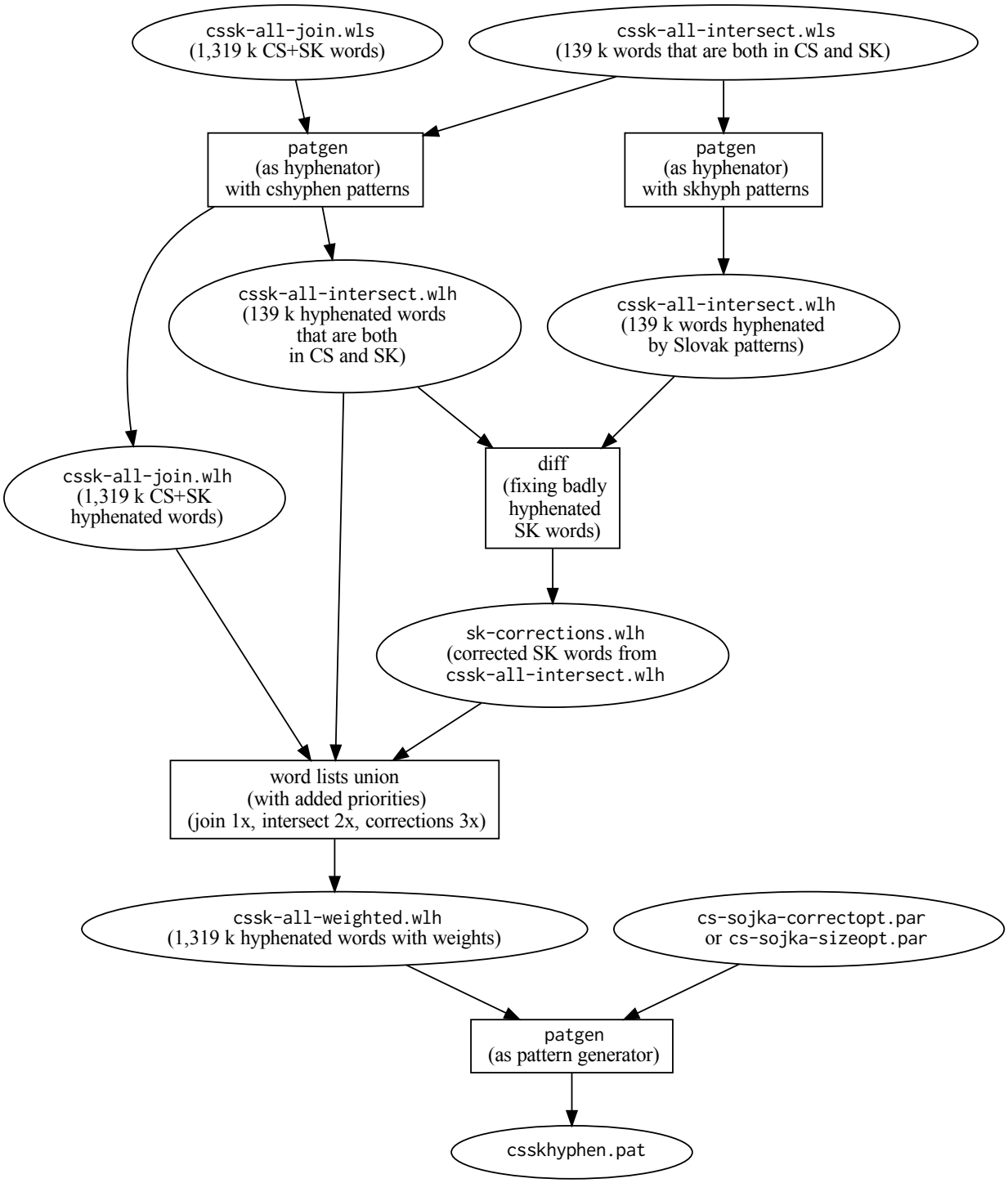

Figure 1: The development process of the new Czechoslovak patterns: Bootstrapping with Czech patterns, checking and fixing with a higher weight Slovak words that are common with Czech ones.

Table 1: Statistics from the generation of Czechoslovak hyphenation patterns with custom parameters.

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

Table 2: Statistics from the generation of Czechoslovak hyphenation patterns with correct optimized parameters.

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

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 e.g. purely Czech patterns.

Word list	Parameters	Good	Bad	Missed	Size	Patterns
Slovak	[4, by hand]	N/A	N/A	N/A	20 kB	2,467
Czech	correctopt [19]	99.76%	2.94%	0.24%	30 kB	5,593
Czech	sizeopt [19]	98.95%	2.80%	1.05%	19 kB	3,816
Slovak	[16, Table 1, `patgen`]	99.94%	0.01%	0.06%	56 kB	2,347
Czechoslovak	sizeopt	99.67%	0.00%	0.33%	40 kB	7,417
Czechoslovak	correctopt	99.99%	0.00%	0.01%	45 kB	8,231
Czechoslovak	custom	99.87%	0.03%	0.13%	32 kB	5,907

Table 4: Results of 10-fold cross-validation with evaluated parameters

Parameters	Good	Bad	Missed
correctopt	99.81%	0.15%	0.04%
custom	99.64%	0.22%	0.14%
sizeopt	99.41%	0.18%	0.40%

We expand on this in the Jupyter notebook [13]. Gentle readers may also find the scripts used there.

5 Evaluation

We evaluated the quality of developed patterns by two metrics. *Coverage* of hyphenation points in the training word list tells how many hyphenation points used in training were correctly predicted by the patterns. *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 as a sum of TP and TN,

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 ten-fold 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 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 word list merging to develop generic patterns for syllable-based 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 [7]

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** [13].

We offer the new patterns for "the Czechoslovak language" to the `hyph-utf8` repository [11] and TeX Live distribution. One might further develop common patterns for additional syllable-based languages to be shared. Then, we will not need precise language mark-up anymore. 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 [19] at TUG 2019, which has led us in this research direction. We also thank all who commented on our workflow, patterns, and this paper.

References

[1] R.E. Allen, ed. *The Oxford Spelling Dictionary*, vol. II of *The Oxford Library of English Usage*. Oxford University Press, 1990.

[2] Anonymous. *The Chicago Manual of Style*. University of Chicago Press, Chicago, 17 edition, Sept. 2017.

[3] S. Bartlett, G. Kondrak, C. Cherry. Automatic syllabification with structured SVMs for letter-to-phoneme conversion. In *Proceedings of ACL-08: HLT*, pp. 568–576, Columbus, Ohio, June 2008. Association for Computational Linguistics. `https://www.aclweb.org/anthology/P08-1065`

[4] J. Chlebíková. Ako rozdeliť (slovo) Československo (How to hyphenate the word Czechoslovakia). *Zpravodaj CS TUG* 1(4):10–13, Apr. 1991. `10.5300/1991-4/10`

[5] P.B. Gove, M. Webster. *Webster's Third New International Dictionary of the English language Unabridged*. Merriam-Webster Inc., Springfield, Massachusetts, U.S.A, Jan. 2002.

[6] M. Jakubíček, A. Kilgarriff, et al. The TenTen Corpus Family. In *Proc. of the 7th International Corpus Linguistics Conference (CL)*, pp. 125–127, Lancaster, July 2013.

[7] M. Keary. On hyphenation – anarchy of pedantry. *PC Update, The magazine of the Melbourne PC User Group*, 2005. `https://web.archive.org/web/20050310054738/http://www.melbpc.org.au/pcupdate/9100/9112article4.htm`

[8] A. Kilgarriff, P. Rychlý, et al. The Sketch Engine. In *Proceedings of the Eleventh EURALEX*

International Congress, pp. 105–116, Lorient, France, 2004.

[9] F.M. Liang. *Word Hy-phen-a-tion by Com-puter*. Ph.D. thesis, Department of Computer Science, Stanford University, Aug. 1983.

[10] Y. Marchand, C.R. Adsett, R.I. Damper. Automatic Syllabification in English: A Comparison of Different Algorithms. *Language and Speech* 52(1):1–27, 2009. 10.1177/0023830908099881

[11] A. Reutenauer, M. Miklavec. TEX hyphenation patterns. Accessed 2021-07-14. https://tug.org/tex-hyphen/

[12] Y. Shao, C. Hardmeier, J. Nivre. Universal Word Segmentation: Implementation and Interpretation. *Transactions of the Association for Computational Linguistics* 6:421–435, 2018. 10.1162/tacl_a_00033

[13] O. Sojka, P. Sojka. cshyphen repository. https://github.com/tensojka/cshyphen

[14] P. Sojka. Notes on Compound Word Hyphenation in TEX. *TUGboat* 16(3):290–297, 1995. https://tug.org/TUGboat/tb16-3/tb48soj2.pdf

[15] P. Sojka. Hyphenation on Demand. *TUGboat* 20(3):241–247, 1999. https://tug.org/TUGboat/tb20-3/tb64sojka.pdf

[16] P. Sojka. Slovenské vzory dělení: čas pro změnu? In *Proceedings of SLT 2004, 4th seminar on Linux and TEX*, pp. 67–72, Znojmo, 2004. Konvoj. https://fi.muni.cz/usr/sojka/papers/skhyp.pdf

[17] P. Sojka. Slovenské vzory dělení: čas pro změnu? (Slovak Hyphenation Patterns: A Time for Change?). *CSTUG Bulletin* 14(3–4):183–189, 2004. 10.5300/2004-3-4/183

[18] P. Sojka, O. Sojka. The Unreasonable Effectiveness of Pattern Generation. *Zpravodaj CSTUG* 29(1–4):73–86, 2019. 10.5300/2019-1-4/73

[19] P. Sojka, O. Sojka. The unreasonable effectiveness of pattern generation. *TUGboat* 40(2):187–193, 2019. https://tug.org/TUGboat/tb40-2/tb125sojka-patgen.pdf

[20] P. Sojka, O. Sojka. Towards New Czechoslovak Hyphenation Patterns. *Zpravodaj CSTUG* 30(3–4):118–126, 2020. https://cstug.cz/bulletin/pdf/2020-3-4.pdf#page=16

[21] P. Sojka, P. Ševeček. Hyphenation in TEX — Quo Vadis? *TUGboat* 16(3):280–289, 1995. https://tug.org/TUGboat/tb16-3/tb48soj1.pdf

[22] N. Trogkanis, C. Elkan. Conditional random fields for word hyphenation. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pp. 366–374, Uppsala, Sweden, July 2010. Association for Computational Linguistics. https://www.aclweb.org/anthology/P10-1038