The Unreasonable Effectiveness of Pattern Generation

Petr Sojka and Ondřej Sojka

Abstract

Languages are constantly evolving organisms, and so are the hyphenation rules and needs. The effectiveness and utility of TEX's hyphenation have been proven by its usage in almost all typesetting systems in use today. The current Czech hyphenation patterns were generated in 1995 and no hyphenated word database is freely available.

We have developed a new Czech word database and have used the `patgen` program to efficiently generate new effective Czech hyphenation patterns and evaluated their generalization qualities. We have achieved almost full coverage on the training dataset of 3,000,000 words and validated the patterns on the testing database of 105,000 words approved by the Czech Academy of Science linguists.

Our pattern generation case study exemplifies an effective solution of widespread dictionary problem. The study has proved the versatility, effectiveness and extensibility of Liang's approach to hyphenation developed for TEX. The unreasonable effectiveness of pattern technology has lead to applications that are and will be used even more widely even 40 years after its inception.

> ... the best approach appears to be to embrace the complexity of the domain and address it by harnessing the power of data: if other humans engage in the tasks and generate large amounts of unlabeled, noisy data, new algorithms can be used to build high-quality models from the data. (Peter Norwig, [7])

1 Introduction

In their famous essays, Wigner [18], Hamming [1] and Norwig [7] consider mathematics and data-driven approaches miraculously, unreasonably effective. One of the very first mathematically founded approaches that harnessed the power of data was Franklin Liang's language-independent solution for TEX's hyphenation algorithm [6] and his program `patgen` for generation of hyphenation pattern from a word list.

Dictionary problem The task at hand was the *dictionary problem*. A dictionary can be seen as a database of records; in each record, we distinguish the key part (the word) and the data part (its division). Given the already hyphenated word list of a language, a set of *patterns* is magically generated. Language hyphenated patterns are much smaller than original word list and typically encode almost all hyphenation points in input list without mistakes. Liang's pattern approach thus could be viewed as an efficient lossy or lossless *compression* of hyphenated dictionary with several orders of magnitude compression ratio.

It has been proved [14, chapter 2] that optimization problem of exact lossless pattern minimization is non-polynomial by reduction to the minimum set cover problem.

Generated patterns have minimal length, e.g., shortest context possible, which results in their *generalization* properties. Patterns could hyphenate words not seen in the learning phase by analogy: yet another miracle of the generated patterns.

Pattern preparation During 36 years of `patgen`, there were hundreds of hyphenation patterns created, either by hand or *generated* by program `patgen`, or by the combination of both methods [8]. The advantage of *pattern generation* is that one can fine-tune pattern qualities for specific usage. Having an open-source and maintained word list adds another layer of flexibility and usability to the deployment of patterns. This approach is already set up for German variants and spellings [5], and was an inspiration for doing the same for the Czech language.

In this paper, we report on the development of the new Czech word list with a free license and complementary sets of hyphenation patterns. We describe the iterative process of initial word list preparation, word form collection, estimation of pattern generation parameters, and novel applications of the technology.

> 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. (Major Keary)

2 Initial word list preparation

As a rule of thumb, the development of new big hyphenated word list starts on small data set first. The experience and outputs from this initial phase, e.g., hyphenation patterns, are applied to the bigger and bigger ones.

Bootstrapping idea As word lists of a well-established language are sizeable, and manual creation of huge hyphenated word list is tedious work, we used the bootstrapping technique. We illustrate the process of initial word list preparation by the diagram on Figure 1 on the following page. We have obtained a hyphenated word list with 105,244 words from the Czech Academy of Sciences, Institute of

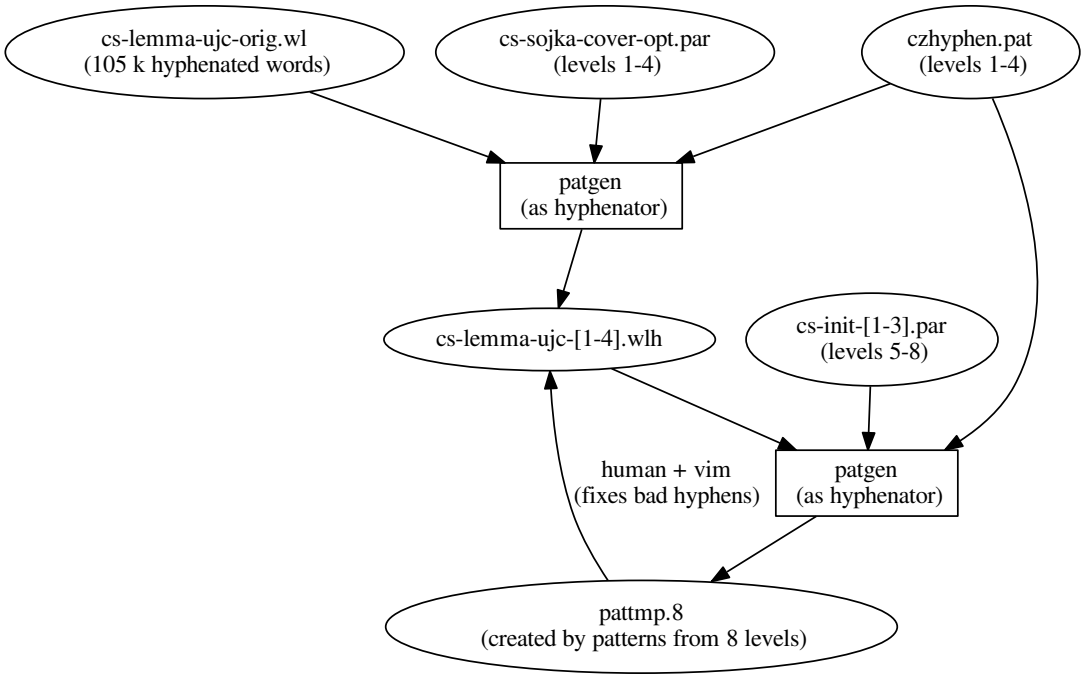

Figure 1: Life cycle of initial word list preparation, illustrated on the development of 105 k Czech consistently hyphenated words. `czhyphen.pat` represents the original Czech hyphenation patterns from [15] and `cs-sojka-cover-opt.par` are correct optimized `patgen` parameters from the same paper. `cs-init-[1-3].par` are custom parameters that trade off bad hyphens (which have to be manually checked) for missed hyphens. Information on which hyphenations `patgen` missed and where it wrongly put a hyphen is sourced from `pattmp`.

the Czech Language (ÚJČ). After a closer inspection, we have discovered many problems with the data, probably stemming from the fact that it has been crafted by multiple linguists and over the years. The few hyphenation rules [2] that are in the Czech language are not getting applied consistently there. The borderline cases were typically between syllabic (ro-zum) and etymological variants (roz-um) of hyphenation, or the way how to handle words borrowed from German or English into Czech.

It is impractical to try and manually find inconsistencies and systemic errors, even in a relatively short word list like this. We slightly modified and extended the process suggested in [13, page 242]: We used `patgen` and the current Czech patterns to hyphenate the word list and manually checked only the 25,813 words where the proposed hyphenation points differed from the official (were bad or missed), creating new word list `cs-lemma-ujc-1.wlh` [11] in the process.

But we are erroneous humans making mistakes. To find these, we have used `patgen` to generate the four additional levels of hyphenation patterns on top of the current patterns from the checked word list. We have also adjusted the parameters, see `cs-init-[1-3].par` [11] used for generation of the four additional levels to trade off bad hyphens (which have to be manually checked) for missed ones. We have then used these patterns, with eight levels in total, to hyphenate the checked word list and manually rechecked the wrongly hyphenated points (dots in `patgen` output), with missed hyphenation point (implicitly marked as the hyphen sign in hyphenated word list). We have repeated this process three times, iterating on `cs-lemma-ujc-[2-4].wlh`. The word list number four, referred to as the 'small one', is used for generation of bootstrapping patterns and final pattern validation.

3 Word list preparation and design

Any live language constantly changes, and Czech is no exception. Many new Czech words now come from

other languages, mostly from English. This presents a challenge for the patterns; they must not only correctly hyphenate Czech words according to Czech syllabic boundaries, but foreign words must be hyphenated correctly too, according to their new Czech syllabic pronunciation. [12] To have the patterns keep up with language evolution, we must maintain not only the patterns, but also a hyphenation word list. In this section, we will detail how we have built such a word list.

csTenTen corpus We have first obtained a word list with frequencies, generated from the Czech Web Corpus of TenTen family (csTenTen) [3]. We then filtered this word list to include only words that are present more than ten times in two crawls [17] made in years 2012 and 2017. We ended up with word list containing 922,216 words, non-negligible part of which are misspellings and jargon.

Word list cleanup We have then cleaned this word list by using the Czech morphological analyzer `majka` [10] to remove all words not known to it. We removed 370,291 typos, misspellings, and similar atypical lexemes and kept only 551,925 frequently occurring valid words in the dataset.

Word list expansion The morphological analyzer `majka` [10] also allows us to expand words into all their flexive forms as shown on Figure 2. We chose not to use the expansion feature of `majka` because the word list would grow to 3,779,379 (almost a fourfold increase) and csTenTen already contains most of the commonly used forms. It would also distort which hyphenation `patgen` gives most weight to. We supply word frequencies from csTenTen to the word list, so one can give higher weight to patterns that cover the most common words, eventually.

We expanded the word list with `majka` by adding 54,569 base forms of words that were present in the word list, but not in their base form. This increased the size to 606,494 words.

abdominální: abdominální, abdominálních, abdominálního, abdominálním, abdominálníma, abdominálními, abdominálnímu, neabdominální, neabdominálních, neabdominálního, neabdominálním, neabdominálníma, neabdominálními, neabdominálnímu

Figure 2: One lemma, expanded into all its lexemes by `majka`.

Sustainability The German *wortliste* [5] project served as inspiration for our open word list format, detailed in the `README.md` [11].

> One must regard the hyphen as a blemish to be avoided wherever possible. (Winston Churchill)

4 Bootstrapping — iterative development of hyphens in the big word list

It would be very tedious to manually hyphenate such a big word list by hand, so we train patterns on the small one and apply them on the big word list, as illustrated in Figure 3 on the following page. Then, we train patterns on the (now hyphenated) big word list and have `patgen` show what it would have hyphenated differently. With this approach, we cherry picks inconsistencies in the word list.

Since the big word list contains not only lemmas (base forms) of words, but also common inflections, we use regular expressions to add hyphens around them and fix inconsistencies. We keep iterating on this, as shown in Figure 3 on the next page, until patterns, generated with `cs-init-[1-3].par` [11], achieve nearly perfect coverage.

The resulting patterns hyphenate according to the standard Czech hyphenation rule: hyphenation is allowed everywhere where it does not change the pronunciation of the word. Thanks to the effectiveness of pattern generation, this works not only in Czech words but also foreign (Latin, French, German, English) ones.

> Hyphens, like cats, are capable of arousing tenderness or shudders. (Pamela Frankau)

5 Pattern generation

The last Czech hyphenation patterns were generated in 1995 [15], and are in use not only in TEX but also in other widespread typesetting systems. For conservative users there is no strong incentive for change, because the error rate is relatively low (the first version of validation set had about 4% error rate), and coverage is relatively high (the first version of validation set had around 7% missed hyphenation points).

Pattern generation from 3,000,000 words doesn't take hours as it did two decades ago, but seconds, even on commodity hardware, which allows for rapid development of "home-made" patterns.

We have developed a Python wrapper for `patgen` that we use in Jupyter notebooks. It allows quick iteration and easy sharing of results — see Table 1 on page 905 or `demo.ipynb` [11].[1]

It has also become common to use a validation dataset to ensure generalization abilities. Our usage

[1] In this preprint version we do not report final Czech hyphenation patterns yet, because we are still iterating on the big wordlist. We will include final statistics in the journal version.

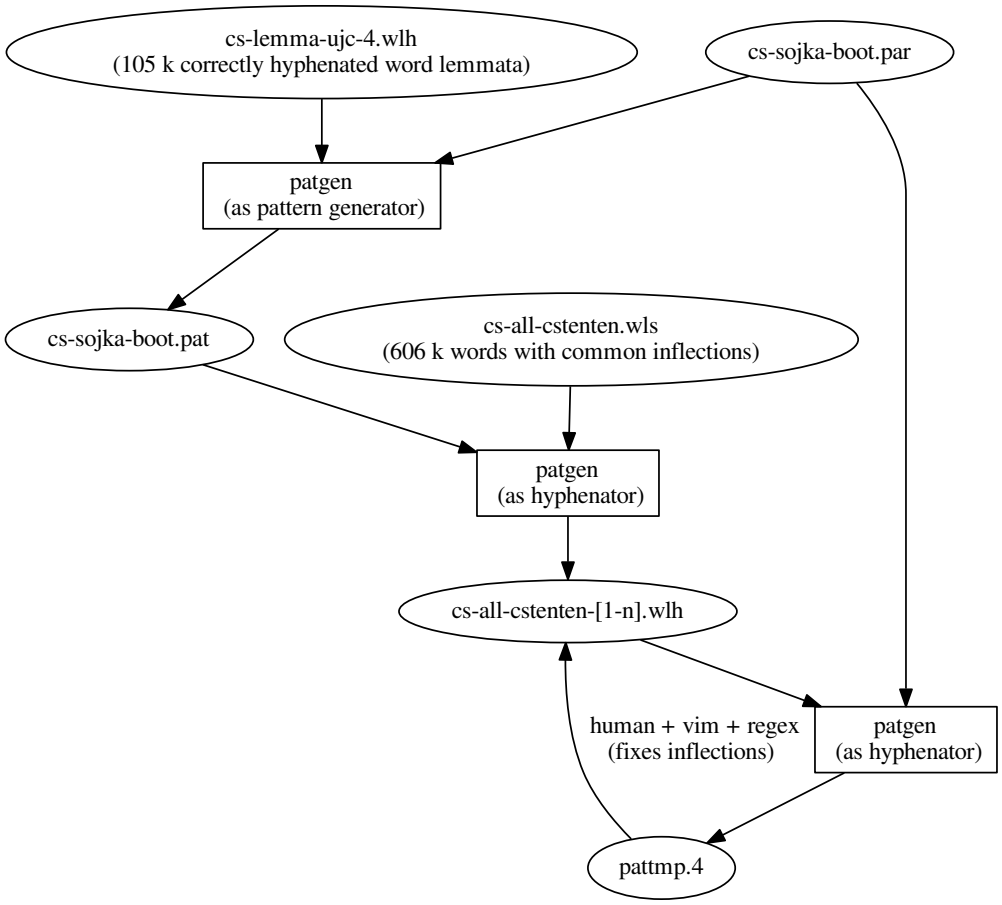

Figure 3: How we bootstrapped hyphenation of the big word list by training patterns (`cs-sojka-boot.pat`) on the small word list and applying them on the big one. `cs-sojka-boot.par` are `patgen` parameters that are designed to generate many patterns but still retain their generalization properties. `pattmp` highlights which hyphenation points in the source file the new pattern level missed, which were correctly covered and where they wrongly put a hyphen.

of a validation dataset has proved useful. Table 2 shows that if we were to use the *correct optimized* parameters from [16] that have been in use for Czech, we would overfit the training dataset and perform *worse* than their *size optimized* counterparts.

We believe that the patterns could be developed and serve as lossless compression of wordlist dataset, thus maximize the effectiveness of pattern technology.

> Life is the hyphen between matter and spirit.
> (Augustus William Hare)

6 The unreasonable effectiveness

We were able to solve the dictionary problem for Czech hyphenation effectively.

Space effectiveness From 3,000,000+ hyphenated words stored in approximately 30,000,000 bytes we have produced patterns of size 30,000 bytes, achieving roughly 1000× space *lossless* compression.

Time effectiveness Using the trie data structure for patterns makes the time complexity of accessing the record related to the word, e.g., hyphenation point, in very low *constant* time. The constant is adequate to the depth of the pattern trie data structure, e.g., 5 or 6 in the case of Czech. In the case, the whole pattern trie resides in RAM, the time for finding the patterns for a word is on the scale of tens, at most hundreds of single processor instructions. Word hyphenation throughput is then about 1,000,000 words per second on a modern CPU.

Table 1: Outputs from running `patgen` in our Jupyter notebook with two different parameter sets. The first parameter set is from the German Trennmuster project [5] and generates 7,291 patterns, 40 kB. The second one from [16] generates shorter and smaller patterns — 4,774 patterns, 25 kB.

Level	Patterns	Good	Bad	Missed	Lengths	Params
1	750	1,683,529	525,670	0	1 5	1 1 1
2	3,178	1,628,874	38	54,655	2 6	1 2 1
3	2,548	1,683,528	9,931	1	3 7	1 1 1
4	1,382	1,683,287	0	242	4 8	1 4 1
5	92	1,683,528	0	1	5 9	1 1 1
6	0	1,683,528	0	1	6 10	1 6 1
7	1	1,683,529	0	0	7 11	1 4 1

Level	Patterns	Good	Bad	Missed	Lengths	Params
1	1,608	1,655,968	131,481	27,561	1 3	1 5 1
2	1,562	1,651,840	2,533	31,689	1 3	1 5 1
3	2,102	1,683,528	2,584	1	2 5	1 3 1
4	166	1,683,135	6	394	2 5	1 3 1

Table 2: A comparison of validation scores of patterns trained on the big (606 k words) wordlist with different parameters.

Params	Good	Bad	Missed	Size	Patterns
correctopt [16]	99.41 %	3.47 %	0.59 %	25 kB	4,774
german [5]	99.40 %	3.33 %	0.60 %	40 kB	7,291

Optimality Even though finding exact space and time-optimal solutions is not feasible, finding an approximate solution close to optimum is possible. Heuristics and insight expressed above, together with Jupyter notebook interactive fine-tuning of `patgen` parameter options allows for rapid pattern development.

Automation A close-to-optimal solution to the dictionary problem could be useful not only for Czech hyphenation, but for all other languages [9, 8], and more generally, for other instances of the dictionary problem. Developing heuristics for thresholding of `patgen` pattern generation parameters, based on a statistical analysis of big input, data could allow the deployment of presented approaches on a much broader problem set and scale. We believe that parameters could be automatically approximated from the statistics of the input data.

Pattern generation — in Wigner terms — "has proved accurate beyond all reasonable expectations". Let us paraphrase another one of his quotes:

> The miracle of the appropriateness of the language of ~~mathematics~~ *patterns* for the formulation of the laws of ~~physics~~ *data* is a wonderful gift which we neither understand nor deserve. We should be grateful for it and hope that it will remain valid in future research and that it will extend, for better or for worse, to our pleasure, even though perhaps also to our bafflement, to wide branches of learning.

> "We should stop acting as if our goal is to author extremely elegant theories, and instead embrace complexity and make use of the best ally we have: the unreasonable effectiveness of data." (Peter Norvig, [7])

7 Conclusion

We have developed a flexible open language–independent system [11] for hyphenation pattern generation. We have demonstrated the effectiveness of this system by updating the old Czech hyphenation patterns [15] and achieving record accuracy. We have also applied recent data and computer science advancements, like the usage of interactive Jupyter notebooks and a validation dataset to prevent overfitting, to the more than three decades old problem of pattern generation.

Future work

Word lists for other languages Logical next steps will be applying developed techniques for different languages: for Slovak and virtually all that does not yet have word list based hyphenation pattern generation and word list either in Sketch Engine or elsewhere are available.

Stratification Pattern generation could be further speed up by several techniques like stratification of word list on the level of input or on the level of counting pros and cons examples to include a new pattern or not.

Pattern-encoded spellchecker We have a big dictionary of frequent spelling errors from csTenTen word list. Nothing prevents us from encoding them into specific patterns or pattern layers with extra levels and used that information during typesetting, e.g., to typeset those words with red underlining in LuaTEX. LuaTEX allows dynamic pattern loading and Lua programming that will enable the implementation of this feature, which people are used to using in editors.

Pattern-based learnable key memories Solutions to versions of dictionary problem are a hot topic of leading-edge research to design memory data architectures like those used in a machine learning of language [4]. Pattern-based memory network architectures could speed up language data access in big memory neural networks considerably.

Acknowledgements

We owe our gratitude to the whole bunch of people: to Vít Suchomel of Lexical Computing for word lists from Sketch Engine, to Pavel Šmerk for `majka` and paper proofreading, to Don Knuth and Frank Liang for TEX and `patgen`, and to Vít Novotný for paper proofreading.

References

[1] R. W. Hamming. The Unreasonable Effectiveness of Mathematics. *The American Mathematical Monthly* 87(2):81–90, 1980. `http://www.jstor.org/stable/2321982`

[2] Internetová jazyková příručka (Internet Language Reference Book). `http://prirucka.ujc.cas.cz/?id=135`

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

[4] G. Lample, A. Sablayrolles, et al. Large Memory Layers with Product Keys, 2019. `https://arxiv.org/pdf/1907.05242`

[5] W. Lemberg. A database of German words with hyphenation information. `https://repo.or.cz/wortliste.git`

[6] F. M. Liang. *Word Hy-phen-a-tion by Com-put-er*. PhD thesis, Department of Computer Science, Stanford University, Aug. 1983.

[7] F. Pereira, P. Norvig, and A. Halevy. The Unreasonable Effectiveness of Data. *IEEE Intelligent Systems* 24(02):8–12, Mar. 2009. `doi:10.1109/MIS.2009.36`

[8] A. Reutenauer and M. Miklavec. TEX hyphenation patterns. `https://www.tug.org/tex-hyphen/`

[9] K. P. Scannell. Hyphenation patterns for minority languages. *TUGboat* 24(2):236–239, 2003.

[10] P. Šmerk. Fast Morphological Analysis of Czech. In P. Sojka and A. Horák, eds., *Proceedings of Recent Advances in Slavonic Natural Language Processing, RASLAN 2009*, pp. 13–16, Karlova Studánka, Czech Republic, Dec. 2009. Masaryk University. `http://nlp.fi.muni.cz/raslan/2009/`

[11] O. Sojka and P. Sojka. cshyphen repository. `https://github.com/tensojka/cshyphen`

[12] P. Sojka. Notes on Compound Word Hyphenation in TEX. *TUGboat* 16(3):290–297, 1995.

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

[14] P. Sojka. *Competing Patterns in Language Engineering and Computer Typesetting*. PhD thesis, Masaryk University, Brno, Jan. 2005.

[15] P. Sojka and P. Ševeček. Hyphenation in TEX — Quo Vadis? *TUGboat* 16(3):280–289, 1995.

[16] P. Sojka and P. Ševeček. Hyphenation in TEX — Quo Vadis? In M. Goossens, ed., *Proceedings of the TEX Users Group 16th Annual Meeting, St. Petersburg, 1995*, pp. 280–289, Portland, Oregon, U.S.A., 1995. TEX Users Group.

[17] V. Suchomel and J. Pomikálek. Efficient web crawling for large text corpora. In A. Kilgarriff and S. Sharoff, eds., *Proc. of the seventh Web as Corpus Workshop (WAC)*, pp. 39–43, Lyon, 2012. `http://sigwac.org.uk/raw-attachment/wiki/WAC7/wac7-proc.pdf`

[18] E. P. Wigner. The Unreasonable Effectiveness of Mathematics in the Natural Sciences. Richard Courant Lecture in Mathematical Sciences delivered at New York University, May 11, 1959. *Communications on Pure and Applied Mathematics* 13(1):1–14, 1960. doi:10.1002/cpa.3160130102

◇ Petr Sojka
The Faculty of Informatics at Masaryk University
Brno, Czech Republic and $\mathcal{CS}$TUG
sojka (at) fi dot muni dot cz
https://www.fi.muni.cz/usr/sojka/

◇ Ondřej Sojka
$\mathcal{CS}$TUG
Brno, Czech Republic
ondrej.sojka (at) gmail dot com