Towards Perfection of Machine Learning of Competing Patterns

The Use Case of Czechoslovak Patterns Development

Ondřej Sojka and Petr Sojka

Faculty of Informatics, Masaryk University, Brno, Czech Republic
454904@mail.muni.cz, sojka@fi.muni.cz

Abstract. Finding space- and time-effective even *perfect* solution to the dictionary problem is an important practical and research problem, which solving may lead to a breakthrough in computation. Competing pattern technology from TEX is a special case, where for a given dictionary a word segmentation is stored in the competing patterns yet with very good generalization quality. Recently, the unreasonable effectiveness of pattern generation has been shown – it is possible to use hyphenation patterns to solve the dictionary problem jointly even for several languages without compromise.

In this article, we study the effectiveness of `patgen` for the supervised machine learning of the generation of the Czechoslovak hyphenation patterns. We show the machine learning techniques to develop competing patterns that are close to being perfect. We evaluate the new approach by improvements and space savings we gained during the development and finetuning of Czechoslovak hyphenation patterns.

Keywords: dictionary problem, effectiveness, hyphenation patterns, `patgen`, syllabification, Czech, Slovak, Czechoslovak patterns, machine learning

"When you're passionate about something, you want it to be all it can be."
Debra Messing

1 Introduction

Dictionary problem is the task of storing a dictionary seen as a database of words where we distinguish the key part (the word) and the data part (values of the key). Finding space- and time-effective even *optimal* solution to the dictionary problem is an important practical and research problem. Solving it may lead to a breakthrough in computation. The effectiveness of the solution lies in the implicit data structures used. Typically some sort of trees (B-trees [1], tries) or hashing or their combination is used [4]. Time complexity is constant

A. Horák, P. Rychlý, A. Rambousek (eds.): Proceedings of Recent Advances in Slavonic Natural Language Processing, RASLAN 2023, pp. 113–120, 2023.

$O(1)$ for both tree-based solutions (constant C is the tree depth to locate values in the list or hash computation time) and space in $O(D)$, e.g. linear in dictionary storage size D. Absolute value of C and linear coefficient for D are important.

In TEX, a solution to the dictionary problem is used for hyphenation. For a given *key*, e.g. a word to be hyphenated, the *values* are the positions of a word where hyphenation may occur. To minimize the storage size of ever-growing dictionaries Frank Liang designed the *competing pattern* technology for TEX [6]. The dictionary problem is decomposed in such a way that word segmentation is stored in the competing patterns generated from the already hyphenated wordlist.

Recently, the unreasonable effectiveness of pattern generation [10] has been shown. It is possible to use hyphenation patterns to solve the dictionary problem even for several languages without compromise. Also, multiple languages could be covered in the same set of patterns [12,7]. All these developments trigger the necessity of effectiveness and of bringing new solutions.

In this article, we show the effectiveness of `patgen` for the generation of the Czechoslovak hyphenation patterns that are close to being optimal.

The paper is structured as follows. We describe competing patterns in Section 2. We define pattern development processes in machine learning nomenclature and define metrics for rigorous evaluation of dictionary problem solutions by competing patterns in Section 3. Section 4 shows an experiment with the hyphenation model development and dataset cleaning. Experiment with grid search of parameter generation and the achieved results are in Section 5. We show that designed techniques and the grid search of parameter generation lead to the development of hyphenation patterns with effectiveness improvements and space savings on the use case of Czechoslovak hyphenation patterns.

Finally, we describe the potential for future work in Section 6 and conclude by Section 7.

> "All fixed set patterns are incapable of adaptability or pliability.
> The truth is outside of all fixed patterns." – Bruce Lee

2 Competing Patterns

Frank Liang [6] designed an efficient solution to a dictionary problem with ***competing patterns***. Patterns are *generated* from the dictionary in the form of an already hyphenated wordlist with program `patgen`. [3]

Generation is decomposed into phases called levels. In each level, all character patterns in the range of length are considered. Patterns added in odd levels are covering, they add new hyphenation points given the letter context, while in even levels and inhibiting, e.g. forbid hyphenation points. Iteration of covering and inhibiting levels creates a hierarchy of exceptions. The patterns generated in odd levels ***compete*** with those generated in even levels whether to hyphenate or not.

The key to having both high-coverage and small sets of patterns with no bad hyphenation point allowed lies in the setting of thresholds for each level

that decide whether patterns will or will not be included in the final set of patterns. [8]

An example of competing patterns generation from the Czechoslovak wordlist of cca 600,000 hyphenated words (8.5 MB) is in [11, Table 2]. The generated pattern dictionary of 8,231 patterns has a size of 45 kB. Patterns loaded into RAM in the packed trie data structure are even smaller, reaching a compression ratio of around 2000:1. The hyphenation value for the input word is found in the constant time of several instructions needed to reach the list of trie storing the pattern.

The competing pattern generation technique thus maps the dictionary problem of storing the hyphenation point for all words of language into the dictionary problem of storing orders of magnitude smaller sets of short patterns.

Another *crucial* advantage of pattern-based solution is that short patterns learn ***hyphenation rules*** that are applicable to words not seen during training. As new words steadily appear in natural languages, learning hyphenation rules rather than hyphenated wordlist brings new ***generalization*** properties.

"In God we trust, all others bring data." — W Edwards Deming

3 Evaluation Metrics

The preparation of patterns from a wordlist is a typical ***supervised machine learning*** solution to dictionary problems.

There are four numbers in the confusion matrix (also called contingency table) 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 the evaluation results, we report several metrics:

Good sum or percentage of found hyphenation points as a **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),

Precision defined as $\frac{\mathbf{Good}}{\mathbf{Good}+\mathbf{Bad}} = \frac{\mathbf{TP}}{\mathbf{TP}+\mathbf{FP}}$,

Recall defined as $\frac{\mathbf{Good}}{\mathbf{Good}+\mathbf{Missed}} = \frac{\mathbf{TP}}{\mathbf{TP}+\mathbf{FN}}$,

F-score, F_β defined as $F_\beta = (1+\beta^2)\cdot\frac{\mathbf{Precision}\cdot\mathbf{Recall}}{(\beta^2\cdot\mathbf{Precision})+\mathbf{Recall}} = \frac{(1+\beta^2)\cdot\mathbf{TP}}{(1+\beta^2)\cdot\mathbf{TP}+\beta^2\cdot\mathbf{FN}+\mathbf{FP}}$, where a positive real factor β is chosen such that **Recall** is considered β times as important as **Precision**.

As **Precision** is much more important than **Recall**, we report $F_{1/7}$ and $F_{1/77}$: type 1 errors are more severe than type 2 errors in our hyphenation points setup.

Nonzero **Bad** or **Missed** 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, the patterns found

an inconsistency that needs to be fixed in the underlying wordlist, rather than a valid exception.

There are two main parts of the machine learning (ML) solution: ***model development*** and ***model evaluation***. The practice of manually inspecting and fixing bad hyphenation points has been used during the ***model development*** of the wordlist so that the data do not contradict each other. **Precision**, sometimes called *coverage*, tells how many hyphenation points used in training were correctly predicted by the patterns.

The ***model evaluation*** of the quality of developed patterns could be done with the same metrics as for the model development of a hyphenated wordlist.

Evaluation of *generalization* properties, e.g. how the patterns behave on unseen data, has to be done on the words *not* available in the data used during `patgen` training. The dataset has to be split into non-overlapping training and validation test sets.

To assess the *generalization* properties, we used 10-fold *cross-validation*, leaving *validation dataset* – one-tenth out of the training set – to evaluate the effectiveness metrics of the patterns on unseen words.

There are other effectiveness metrics that could be measured in the dictionary task:

speed of getting values for the given key (word), and
size size of data structure to store keys (words, patterns).

The above metrics computed for our use case of Czechoslovak patterns are reported in tables 2 and 3. It is clear that when adding 33 bad hyphenation points as full word patterns, the coverage is 99.99% with no error on seen words and only 0.15% error rate on unseen new words.

It has been proven that the task of creating the minimal pattern set is NP-complete [9].

"Pleasure in the job puts perfection in the work." – Aristotle

4 Dataset Consistency for Model Development

Even though the previous results testify to unreasonable effectiveness [10], we have designed a model development task by improving consistency of syllable markup. The rationale is that when inconsistent hyphenation points are marked in the data, more patterns are needed to cover all those idiosyncrasies.

Natural language is in continual development. In Czech and Slovak, some compound words like `roz-um` are no longer considered compounds with hyphenation points separating constituent words. Instead, syllabic hyphenation `ro-zum` is preferred.

Further, syllabic rules hold also near the word border, while it is forbidden to hyphenate so that a single character is cut during hyphenation.

We have semiautomatically filtered 25,273 words that start with one character vowel syllable (`aeiouy`), and added a hyphenation point after it in `patgen`

Table 1: Pattern generation parameters: statistics from the generation of Czechoslovak hyphenation patterns in 2020 [11] with correct optimized `patgen` generation parameters (correctopt2020)

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 2: Coverage and Effectiveness: comparison of the efficiency of different settings to generate Czechoslovak patterns in 2020 [11]

Word list	Parameters	Good	Bad	Missed	Size	Patterns
2020	custom2020	99.67%	0.00%	0.33%	40 kB	7,417
2020	correctopt2020	99.99%	0.00%	0.01%	45 kB	8,231
2020	sizeopt2020	99.87%	0.03%	0.13%	32 kB	5,907

Table 3: Generalization: results of 10-fold cross-validation with evaluated parameters

Wordlist	Parameters	Good	Bad	Missed
2020	custom2020	99.85%	0.22%	0.15%
2020	correctopt2020	99.95%	0.15%	0.05%
2020	sizeopt2020	99.58%	0.18%	0.42%

wordlist. These points are typically filtered out during typesetting by setting of both `hyphenmin` registers to 2. We call the new wordlist dataset model 2023uniqlr1: it comes with slightly changed syllable markup and word deduplication.

The results are provided in tables 4 and 5 on the next page. The change gives better coverage metrics but slightly worse generalizations, probably because of introducing other inconsistencies.

"If I had more time I would have written you a shorter letter."
– Blaise Pascal

5 Parameter Optimization of Pattern Generation

The quality and effectiveness of generating patterns depend on parameters of `patgen` for generation. There is not much insight and heuristics on how to set up `patgen` parameters. The most basic hyperparameter tuning method is setting

Table 4: The effect of consistency: statistics from the generation of Czechoslovak hyphenation patterns with consistent syllable markup added for one character syllables at the beginning of words and `\lefthyphenmin` and `\righthyphenmin` set to `1` (`patgen` generation parameters (correctopt2020))

Level	Patterns	Good	Bad	Missed	Lengths	Params
1	2,675	1,605,899	127,339	61,344	1 3	1 5 1
1	1,505	1,604,012	1,883	63,231	1 3	1 5 1
3	4,289	1,667,204	5,390	39	2 6	1 3 1
4	723	1,666,990	3	253	2 7	1 3 1

Table 5: The effect of consistency on generalization: results of 10-fold cross-validation with evaluated parameters

Parameters	Good	Bad	Missed	Size	Patterns	Precision	$F_{1/7}$
custom2020	99.40%	0.75%	0.60%	29 kB	5,124	0.9925	0.9925
correctopt2020	99.57%	0.83%	0.42%	50 kB	8,384	0.9916	0.9917
sizeopt2020	99.11%	0.72%	0.88%	35 kB	5,955	0.9927	0.9927

Table 6: Parameters found by grid search on wordlist dataset model 2023uniqlr1. Generalization metrics: Good: 99.60%, Bad: 0.86%, Missed: 0.40%, Precision: 0.9914, Recall: 0.9960, F-Score ($\beta = 1/7$): 0.9915, F-Score ($\beta = 1/77$): 0.9914

Level	Patterns	Good	Bad	Missed	Lengths	Params
1	2,216	1,615,261	187,508	51,982	1 3	1 4 1
2	1,726	1,612,057	1,896	55,186	1 3	1 4 1
3	4,198	1,667,198	2,647	45	2 6	1 4 1
4	474	1,667,112	0	131	2 7	1 4 1

a grid search. Grid search is a method to perform hyperparameter optimization, that is, it is a method to find the best combination of hyperparameters. Given the exponential growth of setting combinations, at least hopeful parameter combinations are evaluated.

In tables 6 and 7 we report the best pattern generation parameters found in our limited grid search. By changing the linear factor of the number of bad hyphenation points we achieved our best setup with $F_{1/7}$-scores above .9916.

Table 7: Parameters found by grid search on wordlist dataset model 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 3 1
2	1,872	1,620,864	4,504	46,379	1 3	1 4 1
3	3,886	1,667,204	5,414	39	2 6	1 3 1
4	729	1,666,973	0	270	2 7	1 4 1

"While AI programs try to understand sentences by analyzing word patterns, we try to hyphenate words by analyzing letter patterns." – Frank Liang [6, page 42]

6 Future Work

The feasibility of universal patterns that comprise information for several languages has been confirmed in [7]. Extending Czechoslovak dataset for other Slavic languages, and generating universal Slavic hyphenation is in progress.

A grid search strategy might be found to minimize the size of the pattern set. The success of reduction of the minimal set cover problem to a dictionary problem solvable with competing patterns would lead to the falling of algorithmic barriers [5]. We are trying to find a monotonous ordering of the set of subsets that minimally covers the original set with methods from [2]. Is P=NP?

"But in the endgame of life, I fundamentally believe the key to happiness is letting go of that idea of perfection." – Debra Messing

7 Conclusion

We have studied the possibilities for improvement of machine learning of competing patterns. We have confirmed the necessity of model development and consistency markup in the input dataset. We have shown that techniques like grid search may improve efficiency even further.

We have used the techniques for the development of Czechoslovak hyphenation patterns. The patterns have been deposited on the LINDAT repository `https://lindat.cz`.

Acknowledgement This work has been partly supported by the Ministry of Education of CR within the LINDAT-CLARIAH-CZ infrastructure LM2023062. We are indebted to Don Knuth for the questioning that has led us in this research direction. Firstly, questioned the common properties of Czech and Slovak hyphenation during our presentation of [10] at TUG 2019. Secondly, he mentioned the P=NP problem during his talk at the Faculty of Informatics MU the same year [13].

References

1. Franceschini, G., Grossi, R., Munro, J., Pagli, L.: Implicit B-trees: a new data structure for the dictionary problem. Journal of Computer and System Sciences **68**(4), 788–807 (2004). https://doi.org/https://doi.org/10.1016/j.jcss.2003.11.003, special Issue on FOCS 2002
2. Graham, R.L., Knuth, D.E., Patashnik, O.: Concrete Mathematics. Addison-Wesley, Reading, MA, USA (1989)
3. Haralambous, Y.: A Revisited Small Tutorial on Patgen, 28 Years After. In electronic form, available from CTAN as `info/patgen2.tutorial` (Mar 2021), `https://mirrors.nic.cz/tex-archive/info/patgen2-tutorial/patgen2-tutorial.pdf`
4. Knuth, D.E.: Sorting and Searching, The Art of Computer Programming, vol. 3. Addison-Wesley, third edn. (1998)
5. Knuth, D.E., Daylight, E.G.: Algorithmic Barriers Falling: P=NP? Lonely Scholar, Geel, Belgium (2014)
6. Liang, F.M.: Word Hy-phen-a-tion by Com-put-er. Ph.D. thesis, Dept. of Computer Science, Stanford University (Aug 1983), `https://tug.org/docs/liang/liang-thesis.pdf`
7. Sojka, O., Sojka, P., Máca, J.: A roadmap for universal syllabic segmentation. TUGboat **44**(2) (2023), `https://doi.org/10.47397/tb/44-2/tb137sojka-syllabic`
8. Sojka, P.: Competing Patterns for Language Engineering. In: Sojka, P., Kopeček, I., Pala, K. (eds.) Proceedings of the Third International Workshop on Text, Speech and Dialogue—TSD 2000. pp. 157–162. LNAI 1902, Springer-Verlag, Brno, Czech Republic (Sep 2000). https://doi.org/10.1007/3-540-45323-7_27
9. Sojka, P.: From Minds to Pixels and Back (Habilitation Thesis). Masaryk University, Brno (Apr 2008)
10. Sojka, P., Sojka, O.: The Unreasonable Effectiveness of Pattern Generation. TUGboat **40**(2), 187–193 (2019), `https://tug.org/TUGboat/tb40-2/tb125sojka-patgen.pdf`
11. Sojka, P., Sojka, O.: Towards New Czechoslovak Hyphenation Patterns. Zpravodaj *CS*TUG **30**(3–4), 118–126 (2020). https://doi.org/10.5300/2020-3-4/118, `https://cstug.cz/bulletin/pdf/2020-3-4.pdf#page=16`
12. Sojka, P., Sojka, O.: New Czechoslovak Hyphenation Patterns, Word Lists, and Workflow. TUGboat **42**(2) (2021), `https://doi.org/10.47397/tb/42-2/tb131sojka-czech`
13. Szaniszlo, T.: Dva bloky otázok a odpovedí od Donalda Knutha na FI MU. (Two questions and answers sessions by Donald Knuth at FI MU). Zpravodaj *CS*TUG **30**(1–2), 64–97 (2020). https://doi.org/10.5300/2020-1-2/64