Automatized Generation of Alphabets of Symbols

Serhii Hamotskyi
Faculty of Informatics and Computer Technology
Department of Computer Systems
Igor Sykorsky Kyiv Polytechnic Institute
37, Prosp. Peremohy, Kyiv, Ukraine, 03056
Email: pchr8@anche.no

Abstract—**In this paper, we discuss the generation of symbols (and alphabets) based on specific user requirements (medium, priorities, type of information that needs to be conveyed). A framework for the generation of alphabets is proposed, and its use for the generation of a shorthand writing system is explored. We discuss the possible use of machine learning and genetic algorithms to gather inputs for generation of such alphabets and for optimization of already generated ones. The alphabets generated using such methods may be used in very different fields, from the creation of synthetic languages and constructed scripts to the creation of sensible commands for Human-Computer Interfaces, such as mouse gestures, touchpads and eye-tracking cameras.**

I. Introduction

The need to create writing systems has been with humankind since the dawn of time, and they always evolved based on the concrete challenges the writers faced. For example, the angular shapes of the runes are very convenient to be carved in wood or stone [1]. Alphabets have been created for various purposes, each subject to its own constraints (morse code, shorthand writing, etc.). The rapid increase of available mediums in the recent decades determined the need for many more alphabets, for very different use cases, such as controlling computers using touchpads, mouse gestures or eye tracking cameras.

We started researching this topic by attempting to manually create a system for writing in shorthand (extremely simplified writing used in courtrooms or journalism optimized for writing speed, not readability), since we were not satisfied by existing ones, and then attempting to automate the process. There have been extremely few noteworthy developments in shorthand since the creation of Teeline Shorthand in 1968, and there have been no attempts known to us of using genetic algorithms or any form of automated generation for their creation.

Many approaches for the generation of alphabets have been used, but we are not familiar with a formalized system for their generation. Manually generated alphabets are usually suboptimal. For example, it might be argued that the Latin alphabet favours the writer more than the reader, since it evolved under the constraints of pen and paper, and those constraints are much less relevant in the computer age; readable symbols could be created using much less space. Fonts which try to overcome this limitation exist, for example using dots in the place of strokes. [2] In a similar fashion, many systems do not use the possibilities given by the medium or context (for example multi-touch gestures), electing to base themselves on already existing (and, therefore, familiar ot the user, but suboptimal context-wise) symbols. A formalized framework or software application capable of gathering requirements, generating symbols, grading them on a set of criteria and mapping them to meanings may be able to overcome many of those limitations.

In this paper such a framework is suggested, and a proof of concept using some of the proposed techniques is demonstrated (using the example of generating glyphs for a shorthand system, without mapping them to actual symbols). The analysis of data that is to be conveyed, the generation of symbols, their grading according to certain criteria and the analysis of resulting alphabets is discussed.

II. High-level description of proposed framework

The framework as proposed consists of five basic steps:

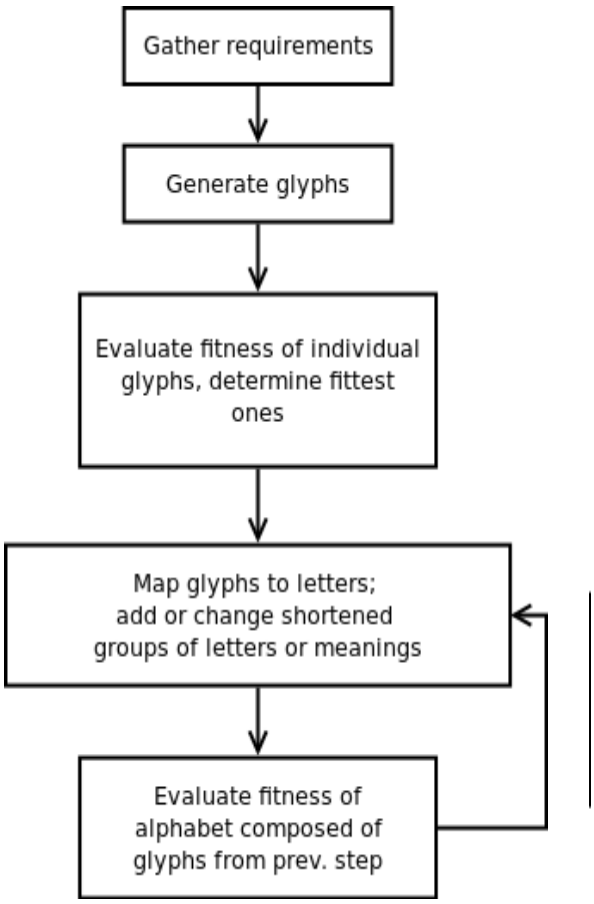

Figure 2: Basic description of proposed framework

III. Gathering requirements for the needed alphabet

As mentioned in the Introduction, most writing systems have been heavily influenced by the constraints inherent in their area of use — purpose, characteristics of the information they needed to convey, materials used. When synthetic systems of symbols are created or generated, they similarly are constrained, but can't be polished by millennia of continuous use. And lastly, even if there *were* millennia to let those systems evolve, even naturally evolving systems tend to converge towards local optima rather than a global optimum (as is the case with genetic algorithms) [3]. Requirements and use patterns may gradually change, while the systems may be stuck in a state that is not optimal anymore, being unable to "sacrifice" short-term fitness for long-term fitness. Therefore, before attempting to create a system from zero, a very careful analysis of the requirements and limitations is needed.

For our example of creating a shorthand system (trivial use case), the following may be considered:

1. On a purely symbolic level:
    1. Writing letters
        1. number of strokes needed to encode individual letters
        2. complexity of the resulting glyphs
    2. Writing words

    1. connections between individual letters (glyphs)
    2. how likely are letters that are easy to connect to each to be represented by easily connectable glyphs
    3. if all existing glyphs are not identical in complexity, what is the ratio of easy-to-write glyphs to the complex ones in a typical text (the bigger the ratio, the better)
  3. Writing sentences:
    1. are there any often-repeating words or groups of words which, when replaced by a shorter, even if complex, symbol, would lead to a gain in time? ("The" as a typical example).
2. On a semantic level**:** Are there any grammatical categories or modalities that are represented in natural text with many letters, that when replaced by a single glyph or a modifier, would lead to a gain in time? (tenses, number, gender, hypotheticals, ...). For example, in our manually generated shorthand system, time, modality, and person were encoded via a single glyph consisting of three parts, which allowed to considerably shorten groups like "they would have been able to".
3. On an information theoretical level: How much redundancy is needed? How many errors in transcription can happen before the message becomes either unreadable or its meaning is distorted? (Natural languages are redundant via multiple mechanisms, notably via agreement in person, gender, case... [4] Errors or interferences will still allow to understand what's being said, up to a certain point. This may not be the case for constructed writing systems, if they are built with low redundancy). [5]

There may be different ways to quantify the above. One way would be **analyzing source texts** written by the person (or representative sample of people) who will be using the system, to fit it to their speech patterns and word use. At the end, the following basic information should be available:

- frequencies of individual letters p_i
- most-needed connections c_{ij}
  Each glyph, in our shorthand example, has three possible starting strokes and three possible ending strokes: $I_s, I_e = \{0,1,2\}$ positioned at different heights. Glyphs such that $i_e = j_s$ are considered easily connectable. Using this information, we generate a list of needed glyphs, such that the letters more likely to be together are represented by glyphs which are easily connectable. This could be trivially solved by having all glyphs start and end at the same height, but this would make more complex differentiating the individual letters (discussed later).

As this is framework for the creation of systems based on individual needs, no final requirements or recommendations are given..

IV. Generation of the glyphs

The second part of the proposed framework is the generation of possible glyphs. In this paper, Bezier curves have been used to generate the glyphs and calculate some of the needed metrics.

Bezier curves are defined by their control points P_0 through P_n, n being their degree. They can be defined for any degree n, the higher the degree the more computationally expensive being their evaluation.

During the generation of the curves, we made the following assumptions about the alphabet for which the glyphs are generated:

1. The glyphs have a definite starting and ending point; the number of such points is limited, to facilitate connecting the symbols to each other.
2. The stroke width does not vary (as, for example, in the case of Pitman shorthand), because of the low availability of pens able to convey even two levels of thickness and of low average penmanship skill in most people. (Though using it as a third or fourth dimension would certainly be possible.)

3. The symbols will fit into a *square* bounding box.
4. During the generation the maximum possible and the average number of control points are set. They have been determined empirically, according to a number of criteria discussed in the following section.

For each letter, multiple glyphs are generated. The generation of glyphs starts by fixing a definite starting and ending point (according to the letter for which we are trying to generate the symbol) and then adding a semi-random number of control points. The number of control points used in the generation of the specific glyph is selected via a normal distribution, with the average number being the mean of the distribution and with standard deviation being calculated based on the maximum number of control points. This has been done because while one of the criteria for the glyphs is their simplicity, extremely simple symbols are counterproductive, and we needed to prevent the alphabet from degenerating to straight and diagonal lines. Additionally, variation in the gene pool is crucial in genetic algorithms, if one is to be used later on. Therefore, a degree of randomness in all phases is needed.

In the first phase of the algorithm, N symbols are generated, each with a normally-distributed number of random control points and start/end in one of the three possible points.

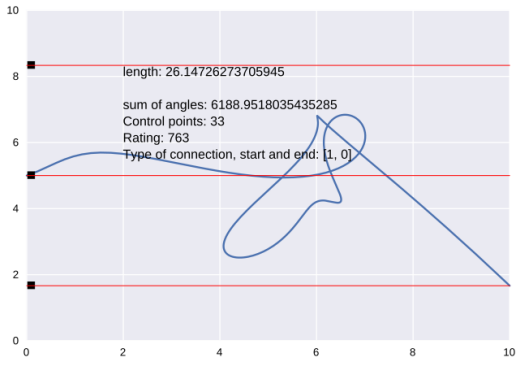

Figure 1: Example of generated glyph with low fitness

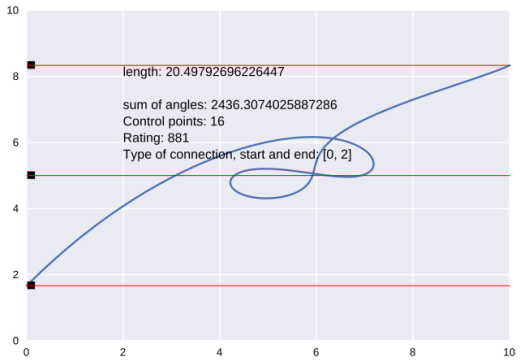

Figure 2: Glyph with higher fitness

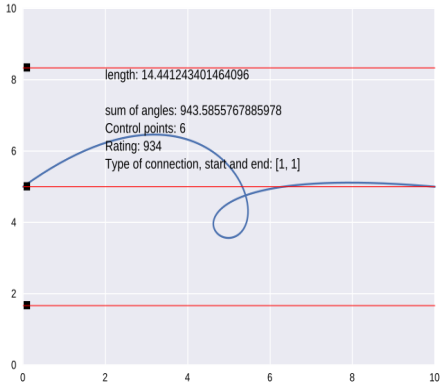

Figure 3: The simpler a glyph is, the higher fitness it has

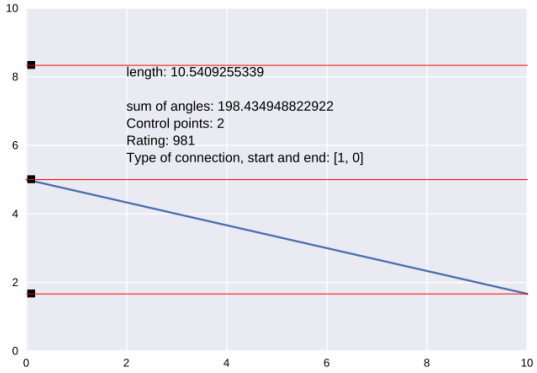

Figure 4: Too many extremely simple symbols would not be desirable

V. Evaluating the fitness of the individual glyphs

In the second stage, the fitness of each glyph is determined. Many approaches are possible, and they heavily depend on the context and the medium for which the generation is being done. For our shorthand system, the main criteria were length and simplicity. The number of control points has been used as a proxy of fitness and has been partly accounted for in the generation phase (empyrically, the more control points the more chaotic the glyph is). The second metric is complexity, which may be loosely defined as "how hard it would be to write this symbol using a pen". This is non-trivial to quantify and requires more research to be done. For our purposes, complexity is defined as $\frac{c}{l}$, where c is the sum of the angles in the polygonal representation of the curve (informally, how curved the glyph is; the more curves there are and the sharper the individual curves are, the bigger the value is), and l is the length of the curve (a certain amount of curves on a large glyph should not be penalized as much as the same amount on a smaller one). C is calculated by converting the curve between the first adjoining control points to a polygon, summing the absolute value of the angles between all adjoining lines, and repeating the process for all the successive control points.

$c = \sum_{i=1}^{n} \sum_{j=2}^{p} L_n(j_i, j_i - 1)$, where n is the number of control points, p ("precision") is the number of lines used to approximate the curve between the control points, L is the angle between two lines, and j_i is line j after control point i. This would bring the formula for fitness to $f = \frac{\sum_{i=1}^{n} \sum_{j=2}^{p} L_n(j_i, j_i - 1)}{l}$.

The reasons for defining c as we did above are manifold, one of them being that a very similar metric is used when evaluating the similarity of two glyphs to each other, as seen in the following section. Much better metrics are possible.

Generally, quantifying fitness of a particular glyph might make use of quite a lot of variables, for example:

- number (percentage?) of strokes which are known to be easy or hard to write (writing a stroke from upper-left to bottom-right might be harder for some people, for example, due to the right slant in their handwriting).

- how easy is a stroke to remember. This might not map perfectly to simplicity, due, for example, to characteristics of human memory like the Von Restoroff effect [6]

The subjective reactions to signs might vary between people, differences due to age, cultural and/or language background are probable. This might be a promising area to study with the help of machine learning — data like "Symbols similar to X perform poorly with demographic Y" would be valuable when creating alphabets when something about the probable users is known (which alphabet or writing system is more familiar to them? How much new information are they probably able to learn and for how much they are able to retain it? Did they learn thousands of hieroglyphics since they were children? How motivated are they?).

Additionally, machine learning would open the doors for custom-tailored systems, where users rate some symbols and based on their feedback predictions are made about what other symbols they might like, remember and use. And, as mentioned previously, their particular use patterns might dictate different mappings of symbols to meanings (letters, actions, preferences).

VI. Mapping symbols to meanings

The first mapping of the generated glyphs, before its fitness is rated, is necessarily very tentative. At the beginning, we suggest just mapping the letters to glyphs by ordering the glyphs in decreasing order of fitness and pairing them with letters, ordered by their frequency. This would give a good starting point, which can be further improved in the next step by taking into account how easy the letters are to connect and the other requirements.

In this paper we have not touched grammatical modalities and ways to shorten them in great detail, as they would merit quite a lot more research and space (and, probably, their own paper); regardless, they would have their place at this step of the framework.

V. Evaluating the fitness of an alphabet

For an alphabet, our goals would be the following:

1. As much high-fitness letters as possible
2. Letters which are found the most often should have the highest fitness (that is, be as simple as possible).
3. The letters should be unlike to each other.

Therefore, one formula for the fitness of an alphabet could be:

$$r = \sum_{i=1}^{n} f_i + \sum_{i=1}^{n} \sum_{j=1}^{n} D_{ij} + \sum_{i=1}^{n} f_i p_i$$

The most important requirement is for the letters to be unlike each other. This is needed both for the resulting text to be readable (the existance of a 1-to-1 mapping between a text written in shorthand and a normal text, or at least for the resulting text being readable using contextual clues) and for improving the memorization of the glyphs (memorizing many similar stimuli is much harder than many different ones, unless a good framework for memorization is given, such as dividing symbols in parts). This leads us to the problem of analyzing how different two given symbols are, and of finding such an alphabet that no two letters are too similar.

For comparing symbols, for our purposes histogram comparison was the most straight-forward to implement. Its main drawback is that it doesn't recognize scaled or rotated images as similar, but glyphs of different sizes and with loops in different places *are* different glyphs. Instead of computing the histogram graphically (pixel-per-pixel), we decided to use the data computed during the evaluation of the fitness of the individual symbols: the angles between the lines of the curves in polygonal form. Basic shapes and turns would be recognizable on this makeshift histogram and, therefore, finding the difference between histograms would be an approximation for the difference between the glyphs they represent.

Out of the many approaches to computing the difference between histograms, Pearson's Chi-squared test has been used. [7]

$$d(H_1, H_2) = \sum_I \frac{(H_1(I) - H_2(I))^2}{H_1(I)}$$

Finally, the glyphs for all needed letters are picked in such a way as to maximize $r = \sum_{i=1}^{n} f_i + \sum_{i=1}^{n} \sum_{j=1}^{n} D_{ij} + \sum_{i=1}^{n} f_i p_i$, the alphabet fitness. The resulting alphabet may then be further optimized using a genetic algorithm, which will result in even further simplified and different letters. Adding/removing/moving control points, switching glyphs between letters, introducing mirror-distortions may be some of the changes the algorithm could attempt.

VII Discussion & future work

The basic ideas of this framework can be applied for the generation of any alphabet used in the real world. For touchpads, for example, connections may be built not using three possible endings, but 2D-points on the screen instead, and multitouch and weight-sensitivity may be included in the generation. By adding dimensions, 3D-gestures alphabets may be created. Much better heuristics for fitness may be created by more precise algorithms, machine learning and use of biology and cognitive science. The approaches demonstrated here are general enough to allow an enormous amount of flexibility in the kind of alphabets they may be used to create.

One of the more interesting avenues of further research would creating algorithms for mapping glyphs to semantics, both letters and more complex grammar categories or structures. This would be the most interdisciplinary out of all the proposed ones. Finding (with AI?) the categories which could be shortened to one or two symbols is challenging by itself, but not all of the possible patterns found by an AI would be intuitive enough for a person to use or even to understand.

VIII Conclusion

In this work in progress paper, a framework for the generation of alphabets is shown. Each of the proposed steps is discussed, and many avenues of future research are found and discussed. Elements of the proposed framework are demonstrated using the generation of symbols for a shorthand writing system. The analysis of source data (in the case of shorthand, sample texts) and requirements as a way to create alphabyets optimal for their use is explored, and the use of machine learning to build better fitness heuristics and to predict the potential fitness of glyphs is discussed. The approach shown may be used to generate alphabets for a variety of methods and purposes.

IX References

[1] Williams, H. (1996). The origin of the runes. Amsterdamer Beiträge zur älteren Germanistik, 45, 211.

[2] Craig Muth. *Dotsies*. Retrieved April 24, 2017, from http://dotsies.org/

[3] Taherdangkoo, Mohammad; Paziresh, Mahsa; Yazdi, Mehran; Bagheri, Mohammad Hadi (19 November 2012). "An efficient algorithm for function optimization: modified stem cells algorithm". Central European Journal of Engineering. 3 (1): 36–50. doi:10.2478/s13531-012-0047-8.

[4] Bussmann, Hadumod (2006). Routledge Dictionary of Language and Linguistics. Routledge. pp. 399–400. ISBN 978-1-134-63038-7

[5] Reza, Fazlollah M. (1994) [1961]. An Introduction to Information Theory. New York: Dover [McGraw-Hill]. ISBN 0-486-68210-2.

[6] Hunt, R. Reed (1995). "The subtlety of distinctiveness: What von Restorff really did" (PDF). Psychonomic Bulletin & Review. 2 (1): 105–112. doi:10.3758/BF03214414.

[7] Pearson, Karl (1900). "On the criterion that a given system of deviations from the probable in the case of a correlated system of variables is such that it can be reasonably supposed to have arisen from random sampling" (PDF). Philosophical Magazine Series 5. 50 (302): 157–175. doi:10.1080/14786440009463897.