Automatic generation of communication grids for people with motor and intellectual disabilities using artificial intelligence methods (Preliminary version)

Carlos Francisco Vargas Figueroa
ENSIMAG and UGA
Grenoble, France
carlos.vargas-figueroa@grenoble-inp.org

Supervised by: Didier Schwab, GETALP - LIG.

I understand what plagiarism entails and I declare that this report is my own, original work.
Name, date and signature: Carlos Vargas Figueroa, 10/06/21, CV

Abstract

Current augmentative and alternative communication (AAC) tools should be assessed with evaluation mechanisms to validate and improve the experience of individuals with communication issues who interact with these systems. Pictogram Grid Communication Systems (PGCS) encodes speech transcripts contained in a corpus into graphical representations of words called pictograms which are organized in a grid pattern.

The aim of this work is to propose and design a method to optimize sentence production in PGCS. The goal is to find the best grid configuration which minimizes the time to build a given sentence of any corpus using classical machine learning algorithms (ML).

This report introduces the context of the problem and provides a description of the approach and the methodology applied. It also presents the current contributions as well as the analysis of the preliminary results of this work. Finally, a brief discussion about new insights to face the problem and future work is presented.

1 Introduction

People with complex communication disorders face big challenges to express themselves in common everyday situations. Augmentative and adaptive communication (AAC) tools are intended to facilitate or enhance their communication skills.

The American Speech-Language-Hearing Association (ASHA) defines AAC as an area of clinical practice that addresses the needs of individuals with significant and problematic communication disabilities characterized by impairments in speech-language production and/or comprehension, including spoken and written modes of communication. [Beukelman *et al.*, 1998]

There are two types of AAC techniques: unaided and aided. Unaided communication does not require any equipment that is external to the body and involves the use of symbols such as manual signs, pantomimes, and gestures. Aided communication incorporates devices that are external to the individuals who use them (e.g. communication book) and involves the use of symbols such as photographs, line drawings, letters, and words [Mirenda, 2003].

One of the aided techniques are Pictogram Grid Communication Systems (PGCS) which are widely used in commercial and non-commercial devices in the market. They are presented in the form of software with speech synthesis on a tablet or in the form of paper files.

This tool has often been created in the field by speech-language pathologists and/or computer scientists, primarily for people with autism spectrum disorders [Mirenda, 2009]. Recent work suggests that the use of these tools with people with complex disabilities (with sensory and motor disorders associated or not with ASD), can be beneficial to the development of communication of these people.

The core of PGCS is a collection of textual documents, called a corpus, which contains transcripts of speech conversations that may include many contexts and situations. The idea is to encode each word of this corpus into a graphical representation called pictogram which can be selected by the user in several ways (touch, gaze tracking, mouse selection) depending on the interface / device. Pictograms belonging to the same semantic space are usually grouped into two-dimensional pages. These pages are connected to other pages through special link pictograms. The final structure can be described as a graph tree with the main (home) page in the root. PGCS adopt different types of vocabulary organization (e.g. lexical, pragmatic, contextual).

The user selects one or multiple pictograms sequentially to build up the desired sentence. The system will provide the corresponding visual and audible output at each step of the process.

In figure 1 we can observe the typical interface of a commercial AAC software application that uses pictograms as tool of interaction.

As any other form of computer human interaction, good design decisions are key in the perceived user experience and adaptability using PGCS. Efficiency in the process of sentence generation is appreciated in this context because it can heavily impact the life quality of end users.

To achieve well suited systems, it is necessary to assess them using evaluation strategies that apply performance met-

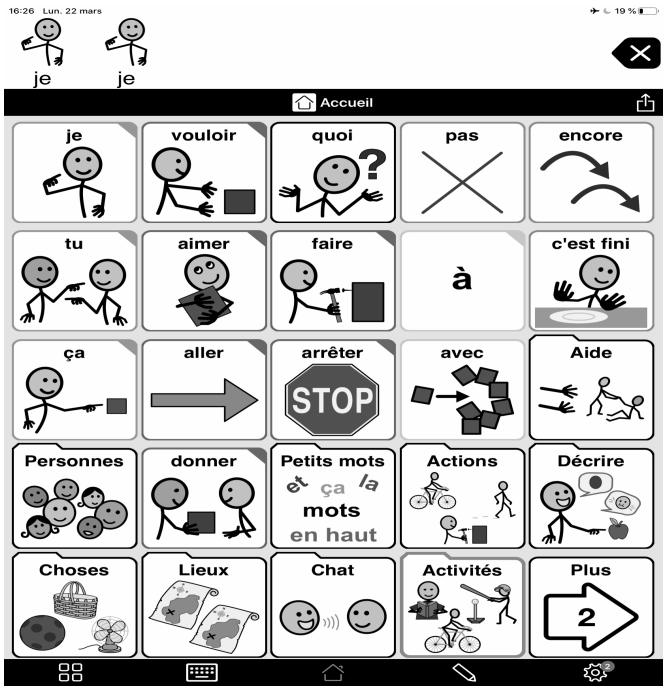

Figure 1: Pictogram-based AAC application. [AssistiveWare, 2021]

rics and optimization methods. A metric gives us an objective way to measure and compare the efficiency of a system, then it is possible to build better systems by minimizing a given cost function.

Unfortunately, these evaluation strategies are quite scarce or non-existent in the current literature, turning its implementation into an empirical task with no benchmark to compare with.

The use of PGCS remains very limited [Moorcroft *et al.*, 2019] for at least two reasons: First, for the person to succeed in using the tool, he or she must be offered a language familiarization process via daily use by the environment, whereas most caregivers expect production use in the short term; Second, caregivers have difficulty using the tools on a daily basis: their acceptability is low [Moorcroft *et al.*, 2019], in particular because use in context is often laborious and requires significant adaptation of caregiver communication.

These limitations are accentuated in France by a delay in the introduction of tools, in particular linked to the persistence of many myths [Cress and Marvin, 2003], but also by the fact that most of the tools currently available are translations from English: it is highly likely that a quasi-literal translation of organizations designed by English speakers for English is not optimal for French.

On the other hand, we have no data on the ability of French speakers (in France) to use these tools or how they can adapt to them with practice. It is clear that the knowledge currently available in automatic language processing, in human-computer interaction and in language sciences is often absent from the design of the tools or very marginally used. In return, the study of the adaptation of typical speakers to this type of device can be considered as an original communication situation to question the adaptive capacities of these speakers, in particular regarding the time-accuracy trade-off, considered here in terms of semantic, lexical and syntactic content but also in sensorimotor terms.

1.1 Hypothesis

Our aim is to conceive a standard method to automatically generate optimal PGCS using a given corpus, this means a grid that offers the lowest production cost for a set of input sentences. To achieve that, we explored a few machine learning based techniques like evolutionary algorithms, which seem to fit the constraints of the problem in question.

A milestone of our project is the implementation of the code that will generate the grids. The proposed steps of the method are:

1. Receive a transcript corpus as input.
2. Create one or multiple grids using a defined structure.
3. Compute the cost of producing a given sentence.
4. Apply a ML-based algorithm to find the best grid candidate.
5. Repeat 3 - 4 until stabilization.

One of the key decisions was to define the type of technique to use for the optimization of the grid generation process. One proposition was to use Neural Networks since it has proved to have incredible potential solving a wide range of problems. Nevertheless, there are several constraints to satisfy: it requires to explicitly define a differentiable loss function having reasonable complexity when computing its gradient. We also need to dispose of ground-truth observations.

One possible workaround is proposed in [Cuturi and Blondel, 2017], it introduces an extension of the dynamic time warping (DTW) discrepancy method for time series. In theory, this approach could be adapted to our problem using generative models. However, due to time constraints, we decided to postpone the exploration of this research branch and focus on more conventional methods that are proved to work with this kind of problems.

We decided to use evolutionary computation which is an example of heuristic search, i.e. search by trial and error, where the 'trials' are candidate solutions, and the 'error' is the measurement of how far a trial is from a desired outcome. The error is used to select which trials are to be used to generate further trials. The fundamental rule-of-thumb is that the best chance to further reduce the error is to generate new trials by making modifications to the previous trials that had the lowest errors [Altenberg, 2016].

In general, evolutionary algorithms (EA) follow the iterative process shown in figure 2.

An EA (e.g. genetic algorithm) assumes that the solution to the proposed problem can be found in a pool of candidate solutions, which is called the search space. It also needs the definition of a cost (objective) function that allows us to measure how far a given solution is from an optimal (or sub-optimal) answer. In the initialization step, the algorithm takes a population of known candidates as input, in this case the

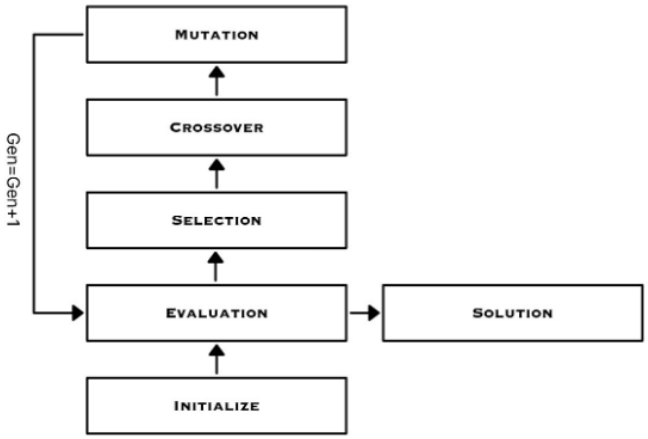

Figure 2: Typical workflow of EA. [Altenberg, 2016]

candidates would be different grids. Variation operators randomly modify these candidates to produce new individuals. The classical variation operators are mutation and crossover (i.e. recombination), based on Mendelian genetics and thus called 'genetic operators'. Mutation is implemented by altering the state of symbols in a string, such as flipping a bit in a binary string from 0 to 1 or vice versa. Crossover is implemented by recombining parts of one parent with parts of another [Altenberg, 2016]. The task of picking the right candidates that will pass to the next generation and will transfer their properties to the new candidates is done in the selection phase. Usually there is a threshold that discriminates the possible candidates based on their individual score computed using the cost function, this is done in the evaluation phase.

The process described in figure 2 is repeated iteratively until the evaluation step yields a desired result or until the error converges to some value and gets stable.

In the next sections we will present a precedent related work, then the current status of this research including the proposed approach and the progress done. Finally, we show the plan to follow for the remaining work to do.

2 Methodology

2.1 Previous Work

Our research initiative is the extension of a previous effort to provide meaningful solutions to the problem described above. The results of that work [Chasseur *et al.*, 2020] include the conception of an open format to encode communication grids. Another achievement was the implementation of one method to evaluate the time production cost of a sentence in a communication grid created via the encoding open format. The cost is defined by two main interactions: selection interaction and displacement interaction. The first one is the time a user takes to execute the selection of a pictogram, whereas the second one is the time to move from one pictogram to another one. These costs depend on individual capabilities and vary from user to user. However, the displacement interaction is also affected by the physical distance between two pictograms, which implies that the structural design (i.e. the position of each pictogram with respect to others) of the grid determines the overall time to compose a sentence.

To calculate the cost for any phrase, the first action is to compute the distance between every pair of pictograms inside every page of the grid, this is done by the function distance calculation. Then, using these distances, the production cost method will represent the grid as a directed graph. Its nodes are the pictograms and the edges are the links among them. The weight of the edges is the cost of navigating between the nodes that compose each edge. Finally, the shortest path linking the words of a sentence is found applying the Dijkstra algorithm.

2.2 Process

The starting point of the work was getting familiar with the PGCS. We performed an extensive review of technical papers and articles to gain insight on the context of the problem. We focused on a previous work [Chasseur *et al.*, 2020] due to its importance for the next stages of the research. As explained before, that project facilitates our research as it provides an initial way to obtain the cost function for a given sentence. However, one of the concerns about this was its own performance, that is, the time the cost function takes to produce a cost estimation. Indeed, after some exploration of the code behind the method, we realized it was highly inefficient. For instance, for the computation cost of a small sentence composed by four basic words, the algorithm took more than three minutes to execute. This supposes a problem since the search space for an optimal grid is usually huge, meaning this function will be used extensively in further stages. For efficiency and environmental purposes, we decided to perform an optimization on it.

Another big step of the project is to define the format of the grids, that is, the implementation of the basic data structures, the graphical representations and the functions to manipulate and mutate the grids. For this we developed an object-based framework in python using Jupyter notebooks. This environment allows to quickly test and run every functionality.

The most basic element of the framework is the slot, which represents either an empty space or a pictogram. It contains information about the word, and the destination of a given pictogram. The next element in the hierarchy is the page, which is composed of slots and defines the maximum number of pictograms grouped at once. Finally, the most general object is the grid. It contains one or several pages and defines the pictogram system.

Our framework has a method to read directly from the Augcom format that we mentioned previously and generate a grid from it. The Augcom format is a text file (.csv) that encodes every pictogram of the grid, one per line, and has relevant information as the word, the position coordinates (row, column) in the page, its container page and the unique identifier formed by the word and the container page. If existed, the destination page is also presented in an independent line, using the identifier and the target page. The destination page attribute allows jumping from the current page to another one, thus connecting every possible pair of pictograms through a path. Listing 1 illustrates the Augcom format.

```
I          1 1      home        i@home
HOUSE      3 1      home        house@home
BUS        1 2      home        bus@home
SPEAK      2 2      home        speak@home
MEALS_D    3 2      home        meals_d@home
                    meals_d@home        meals
STEAK      1 1      meals       steak@meals
PIZZA      2 2      meals       pizza@meals
PASTA      3 3      meals       pasta@meals
```

Listing 1: Format Augcom

Moreover, we developed a fusion strategy that enables the concatenation of two different grids to form a new one with a new internal structure. This is probably the most important feature of the framework, as it provides the flexibility that our system needs to change and eventually converge to an optimal solution. It will be used in the crossover step of the evolutionary algorithm.

Fusion Approach

As our grid structures are organized in a graph fashion, the natural entry point to perform a fusion between two different grids is the 'home' page of each one. The idea behind the fusion method is to combine the pictograms of each grid using some stochastic process, so that the resulting grid contains all the information of the 'parents'. It implies that the semantic relation between a pictogram and its container page might not be maintained after the fusion. In this case, the priority of the method is to allow the construction of new distribution patterns inside the grid and then analyze their performance.

The initial approach we used to implement the fusion consisted in clipping the two home pages of each grid by creating linking pictograms between them using one empty slot on each side. If there were no empty slots in one or both home pages, an additional page was created through a 'splitting' method that redistribute the pictograms of the concerned page into two pages, the original and the new one. Basically, this method only modifies the home pages, keeping intact the structure of the underlying pages beyond those, but integrating them into a new grid.

Although this approach seem to be simple and consistent, it does not allow us to explore the different possible ways to arrange the pictograms between the two grids, making the model rigid and constrained. We decided to change the approach to a one more aligned with our requirements.

The resulting strategy works as follows: Each page of one grid is cross-linked with its counterpart of the second grid. Here we assume that the size of the pages of each grid are the same. The name of a page in the final grid is also randomly selected and if one of the pages does not exist then the other is kept as it is. Each slot is compared with its counterpart at the same position in the other page. The decision of which slot will remain in the resulting grid is random if both slots exist (i.e. they are no empty), otherwise the only existent slot will be selected.

Our intention is to replicate this behavior among all the paired pages to propagate the crossing along the entire structure starting from the home pages until the pages on the bottom of the hierarchy. You have may notice that in the crossing process we only keep at most 50% of the whole information (i.e. pictograms) assuming both pages are fully populated. The remaining pictograms that were not selected will be stored and added at the end of the process in the empty slots of the pages of the resulting grid. If there is no more room, an additional page is created and appended to the grid.

In figure 3 we observe a diagram that exemplifies the operation described above. It presents two grids (g_1 and g_2) being merged, both of them have one home page and one 'child' page (pageX) connected through the pictogram located in the bottom left-most corner (I and R). The home page of the resulting grid is formed taking randomly one of the two options for each slot of the page, which are indicated with the dashed arrows. To have a clean overview, only the diagonal slots were marked with dashed arrows, but all the slots of both pages are compared in this way (e.g. B and K, C and L, etc). In the next level of the hierarchy, we perform the same operation using analog pages, therefore, the 'child' page of the resulting grid is composed by the fusion of pageX(g_1) and pageX(g_2).

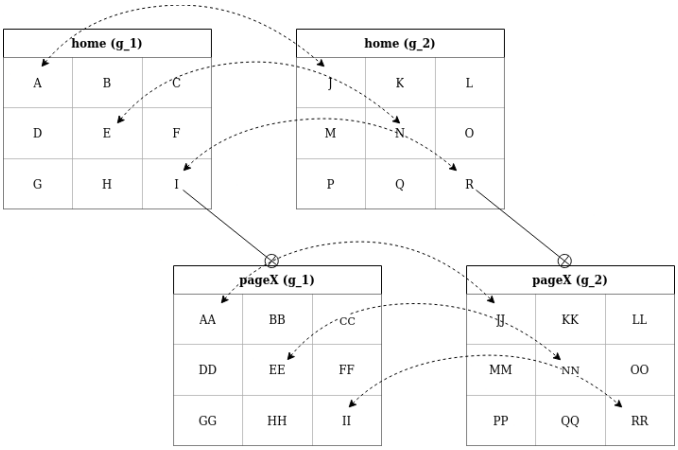

Figure 3: Recursive fusion at each level of the grid hierarchy.

3 Contributions

In the first part of the optimization process, we discovered unnecessary redundant computation in the graph generation step of the production cost function. It was accordingly modified and tested to ensure that the functionality remained intact. In a second review, we realize it could be optimized even more by storing the graph data in memory, avoiding more unnecessary computations. We updated the source code in the web repository with the permission of the authors to include all the optimizations. The results are presented in the analysis section of this paper.

Concerning the progress achieved with the fusion of grids, we present a basic example to illustrate its behavior. We remain that this a critical step of a generic evolutionary algorithm.

We consider two grids 1 and 2 (size 3 x 4), composed only by one page each. The home page of both is shown in figure 4, whereas the resulting grid (grid 3) after the fusion of the

first two is shown in figure 5. The background color of the pictograms in the initial grid is different so that we can easily observe the origin of them in the new grid. The pictograms that were not selected for the home page are stored in the appendix page of grid 3.

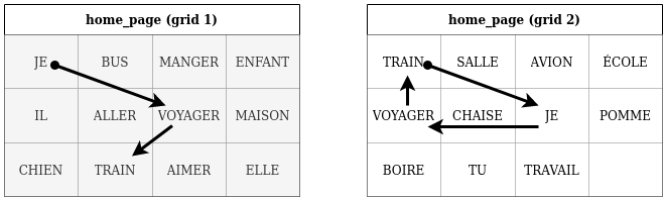

Figure 4: Home page of both grid 1 and grid 2.

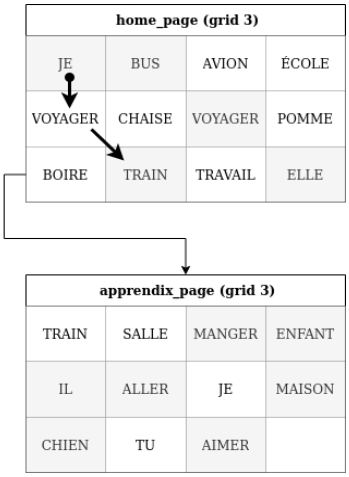

Figure 5: Resulting grid after fusion of grid 1 and grid 2.

To evaluate the grids, we use the *distance calculation* and *production cost* functions developed in a previous work, the last one provides the actual estimation of the time taken to produce a given sentence on a PGCS. For further information and details about the implementation, you can refer to [Chasseur *et al.*, 2020].

Note that we need to set an input phrase for the evaluation, so in this example we use the sentence *"je voyager train"* which literally translates to "I travel train". Although this expression is not syntactically correct, it is usual to have this kind of composition in users with communication issues, specially if they are trying to decrease the latency of the communication process. Thus, the correct meaning of the phrase is "I travel in train", however those syntactic details are not relevant for the purposes of this work.

In this example, the whole phrase is contained in one single page at the beginning, but in reality the words forming a sentence can be very scattered in the structure of the grid, resulting in big production costs even with short and basic sentences as in this case.

We performed one round of computation to simulate, in a simplistic way, one iteration of the process followed by a genetic algorithm, which will be implemented in later stages of this project. We then computed the cost for each of the three grids using the input phrase and compared the score of the initial ones with respect to the new one.

We repeated this procedure several times to get an idea of the improvements we can obtain through the implicit randomness. The results are discussed in the next section.

4 Analysis of Results

As this work has different parts, we will start showing the results obtained in the optimization stage. After solving the issues described above, we got a significant reduction of the execution time of computing a cost. For instance, using a four-word sentence, we passed from about 3 minutes 20 seconds to 1.7 seconds for the whole computation. These results are summarized in figure 6, which details the efficiency gains obtained in each round of improvement.

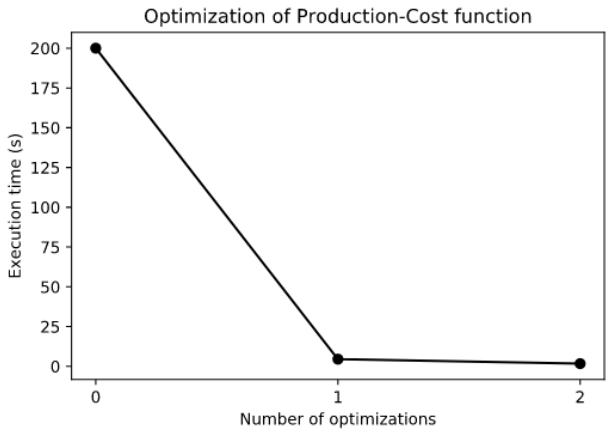

Figure 6: Reduction of execution time by optimization.

Regarding the experimentation with the fusion method, we got the following results: For grid 1 and grid 2 we got an estimated time of 6.65 and 8.23 seconds respectively, whereas for the grid 3 (result of fusion) we got a time of 5.41 seconds. To understand these numbers, we can observe the arrows in figures 4 and 5. They describe the shortest path to build the input sentence, always starting from the top left-most pictogram. As you can see, after the fusion we got a candidate grid that reduced the production time for the given phrase. This individual example does not have any validity in the real word, because it is just one iteration of a simple model, with a very limited initial population and only tested for one single sentence. However, it serves as an illustration of the mechanism we need to implement to solve our problem. It also proves that, in the short term, our approach is able to lead to more optimal solutions.

5 Conclusion

AAC techniques are gaining more and more traction over the years thanks to the effort of many researchers that spend their time improving and adapting these systems to fit real user needs. Another important factor that promote their use is the

democratization of the digital world, which means that today it is easier to get access to quality software solutions and the price of hardware components is constantly decreasing.

This work start with a review of the definition and implications of AAC in the context of people with cognitive and motor limitations. We propose a strategy to generate optimal pictogram grids using a genetic algorithm. We take a previous related work as a starting point and optimize it to get acceptable execution times. Then, we implement the logic of a variation operator which is the core of the genetic algorithm and permit the generation of new grid candidates. Finally, we demonstrate its operation and discuss the results, the limitations and the perspectives of the work to do.

5.1 Future Work

This work is part of an ongoing project and still have a few technical challenges to overcome. For instance, we currently work on the maintenance of the connectivity among the pages after a fusion, since the modifications carried out in this process often break the structural relations among them. Having the fusion operator fully completed, the next step is to develop a custom genetic algorithm which will control the iterative loop. For this, we expect to use the Distributed Evolutionary Algorithms in Python (DEAP) library [Kim and Yoo, 2019], a specialized tool that offers a convenient framework to create and compare genetic models.

The testing strategy begins with simple and basic cases to assess the operation, then we expect to use bigger grids with real corpus to validate the results. The scope of the tests will depend on the available resources. We will aim at a scientific publication at the end of the research project.

References

[Altenberg, 2016] L. Altenberg. Evolutionary computation. In Richard M. Kliman, editor, *Encyclopedia of Evolutionary Biology*, pages 40–47. Academic Press, Oxford, 2016.

[AssistiveWare, 2021] AssistiveWare. *Proloquo2Go - AAC app with symbols*. Amsterdam, Nederland, 2021.

[Beukelman *et al.*, 1998] David R Beukelman, Pat Mirenda, et al. *Augmentative and alternative communication*. Paul H. Brookes Baltimore, 1998.

[Chasseur *et al.*, 2020] Lucie Chasseur, Marion Dohen, Benjamin Lecouteux, Sébastien Riou, Amélie Rochet-Capellan, and Didier Schwab. Evaluation of the acceptability and usability of augmentative and alternative communication (acc) tools: the example of pictogram grid communication systems with voice output. pages 1–3, 10 2020.

[Cress and Marvin, 2003] Cynthia J Cress and Christine A Marvin. Common questions about aac services in early intervention. *Augmentative and Alternative Communication*, 19(4):254–272, 2003.

[Cuturi and Blondel, 2017] Marco Cuturi and Mathieu Blondel. Soft-dtw: a differentiable loss function for time-series. In *ICML*, pages 894–903, 2017.

[Kim and Yoo, 2019] Jinhan Kim and Shin Yoo. Software review: Deap (distributed evolutionary algorithm in python) library. *Genetic Programming and Evolvable Machines*, 20(1):139–142, Mar 2019.

[Mirenda, 2003] Pat Mirenda. Toward functional augmentative and alternative communication for students with autism. 2003.

[Mirenda, 2009] Pat Mirenda. Promising innovations in aac for individuals with autism spectrum disorders. *Perspectives on Augmentative and Alternative Communication*, 18(4):112–113, 2009.

[Moorcroft *et al.*, 2019] A Moorcroft, N Scarinci, and C Meyer. A systematic review of the barriers and facilitators to the provision and use of low-tech and unaided aac systems for people with complex communication needs and their families. *Disability and Rehabilitation: Assistive Technology*, 14(7):710–731, 2019.