Combinatorics vs Grammar: archeology of computational poetry in *Tape Mark I*

Alessandro Mazzei[+] and **Andrea Valle**[*]
[+]Dipartimento di Informatica [*]Dipartimento di Studi Umanistici
CIRMA: Centro Interdipartimentale di Ricerca Sulla Multimedialità e l'Audiovisivo
Università degli Studi di Torino
[alessandro.mazzei|andrea.valle]@unito.it

Abstract

The paper presents a reconstruction of the automatic poetry generation system realized in Italy in 1961 by Nanni Balestrini to compose the poem *Tape Mark I*. The major goal of the paper is to provide a critical comparison between the high-level approach that seems to be suggested by the poet, and the low-level combinatorial algorithm that was actually implemented. This comparison allows to assess the relevance of how the available technology constrained and shaped the work of the poet, to reveal some of his aesthetic assumptions, and to discuss some aspects of the relation between human and the machine in the creative process.

1 Introduction

Systems for automatic poetry generation (APG) introduce specific features when compared to systems for natural language generation (NLG). While most of data-to-text NLG systems usually follow a pipeline architecture (Reiter, 2007), a number of different architectures and techniques have been applied in APG (Gervás, Pablo, 2015). A crucial difference between APG and NLG is the nature of the input, that unavoidably involves its evaluation. The evaluation of a NLG system can be based on a reference corpus, on human evaluation or on the execution of a given task. However, all these evaluation strategies rely on the reception/comprehension of the message, that is, on the meaning units contained in the input. In contrast, in the case of APG, a clear notion of input content is not clearly available, and the evaluation of the output is an opaque task, as it depends on aesthetic (or more largely, cultural), widely variable assumptions. In this sense, APG are similar to other context-evaluated linguistic phenomena such as metaphors. An example of quantitative evaluation of APG based on human judgments is reported in (Toivanen et al., 2012).

By following the classification proposed in (Gervas, 2016), there are two main categories of APG systems: the first category is composed by systems that reuse fragments of text from other poetic texts; the second category is composed by systems that generate a stream of text by using some procedures that exploit word-to-word relations. APG systems from both these categories may use different kinds of linguistic information since fragments fusion, as well as word-to-word relations, can be based on lexical, morpho-syntactical, semantic, rhetorical or metrical theories. Indeed, fragments fusion can be modeled as a string-based fusion in relation to some combinatorial procedure or, in alternative, as a more complex grammar-based fusion, in this case accounting for more sophisticated linguistic theories. Only the detailed analysis of a certain specific APG system, rooted on a reproducible implementation, i.e. algorithms and data structures, can help us to understand the real linguistic creative nature of the poetic generation process involved. Another important component in the analysis of the creative aspects of an APG concerns the non-algorithmic contribution that the poet-programmer may introduce in the final version of the poetic artwork. Indeed, often the poet-programmer, especially in the earlier years of APG, modifies the output provided by the APG system in order to solve some linguistic issues of

Proceedings of the INLG 2016 Workshop on Computational Creativity and Natural Language Generation, pages 61–70,
Edinburgh, September 2016.

the system, or in order to select one among various possible outputs (Funkhouser and Baldwin, 2007).

The aim of this paper is to perform an experiment in *archeology of multimedia* (Lombardo et al., 2006): we first analyze, and then reproduce, the poem *Tape Mark I* by strictly following the actually implemented algorithm (Balestrini, 1962). *Tape Mark I* was a pioneering example of APG dating from 1961, implemented in the assembler of an IBM 7070, one of the first commercial fully transistorized computer. By reproducing the original algorithm we have been able to understand: (1) the details of the creative process related to the combinatorial fusion of textual fragments; (2) the real contribution given by the human poet to the final version of the artwork.

The rest of the paper is organized as follows: Section 2 historically introduces *Tape Mark I*; Section 3 and Section 4 report the computational descriptions of the artwork from, respectively, a high- and low-level perspective; Section 5 describes a simulation experiment regarding the poem; Section 6 critically considers the relation between the author and the algorithm; Section 7 adds some critical conclusions on the evaluation of the system.

2 Balestrini and computer-generated poetry

Funkhauser has reconstructed a chronology of the first attempts in computer-generated poetry (Funkhouser and Baldwin, 2007):

1959: Theo Lutz (a student of the theorist of Information aesthetics Max Bense) implements the first programs for computing poems, *Stochastische Texte* (a text generator);

1960: the French *Oulipo* group is founded (including the notorious writers Queneau and Perec, but also mathematicians, that were mostly concerned with combinatorial approaches from an anti-lyrical perspective);

1960: Brion Gysin composes his permutation poem *I am that I am*, programmed by Ian Somerville;

1961: Nanni Balestrini produces *Tape Mark I* on an IBM 7070;

1961: Rul Gunzenhäuser composes *Weinachtgedicht* (automatic poems).

Thus, *Tape Mark I* is one of the first examples of the use of a computer to generate poetry (Balestrini, 1962; Balestrini, 1968). In 1961 Balestrini (born 1935), while still a young poet, was already a fundamental figure in the Italian avantgarde movement. His poems were included in the crucial anthology *I novissimi* [the newest] (Giuliani, 1961) and he later became a member of the experimental collective *Gruppo '63* (Alicicco et al., 2010). In his long and still continuing career, he has also been the recipient of a prize at Venice Biennale for his visual work, still related to the manipulation of language. Since its inception, Balestrini's work provided a specific version of the main aesthetic assumption theorized by *novissimi* –language as a material reality on which the poet operates– through the extensive manipulation of textual fragments by other authors, retrieved from disparate sources, e.g. novels, essays, poetry, newspapers, popular magazines. This kind of technique, that can be associated to the cutup processes by W. Burroughs and to other collage-based avantgarde approaches (Renello, 2010), was at the basis of *Tape Mark I*, and was to be developed further by the author, leading him to write an entirely computer-generated novel, *Tristano* (1966, (Balestrini, 2016)). The relevance of the poem was immediately recognized internationally as *Tape Mark I* was featured in the first exhibition dedicated (1968) to electronic and computer art, *Cybernetic Serendipity* (Reichardt, 1968; MacGregor, 2002; Boden, 2015).

3 *Tape Mark I*: high-level model

Tape Mark I was included and extensively documented in the *Almanacco Bompiani 1962*, a yearly publication by Bompiani publisher since 1925, that in that year issued a special volume dedicated to the "application of computers to moral sciences and literature" (Morando, 1962). The contribution by Balestrini featured the poem, a description of the generative procedure, information on the implementation and the relative outputs.

Tape Mark I starts from three fragments ("groups") extracted from the following texts (here we report the Italian titles of the first two):

1. *Diario di Hiroshima* by Michihito Hachiya
2. *Il mistero dell'ascensore* by Paul Goldwin
3. *Tao te King, XVI*, by Lao Tse

Figure 1 shows the three groups. Each line is an "element" subdivided into "metrical units" (/) and provided with one "head code" (*codice di testa*) and one "tail code" (*codice di coda*), respectively on the right and left side. Numbers are not to be read as fractions, rather they represent couples of input and output codes. The *Almanacco* provides a set of four

Testo

Codice di testa		Codice di coda
	I (da *Diario di Hiroshima* di Michihito Hachiya)	
1/4	l'accecante / globo / di fuoco	2/3
1/2	si espande / rapidamente	3/4
2/3	trenta volte / piú luminoso / del sole	2/4
3/4	quando raggiunge / la stratosfera	1/2
1/3	la sommità / della nuvola	2/3
2/4	assume / la ben nota forma / di fungo	3/4
	II (da *Il mistero dell'ascensore* di Paul Goldwin)	
1/4	la testa / premuta / sulla spalla	2/4
1/4	i capelli / tra le labbra	2/4
2/3	giacquero / immobili / senza parlare	2/3
3/4	finché non mosse / le dita / lentamente	1/3
3/4	cercando / di afferrare	1/2
	III (da *Tao te King*, XVI, di Laotse)	
1/2	mentre la moltitudine / delle cose / accade	1/2
2/3	io contemplo / il loro ritorno	3/4
1/2	malgrado / che le cose / fioriscano	2/3
2/3	esse tornano / tutte / alla loro radice	1/4

Figure 1: Source text organization in groups.

"instructions" that allow to generate *Tape Mark I* (and, in the poet's ideas, possibly other poems) from the groups (Balestrini, 1968):

I. Make combinations of ten elements out of the given fifteen, without permutations or repetitions.
II. Construct chains of elements taking into account the head-codes and tail-codes
III. Avoid juxtaposing elements drawn from the same extract (i.e. group).
IV. Subdivide the chains of ten elements into six lines of four metrical units each.

The algorithm reported the instructions in natural language without any specifications about the data structures. Step I introduces a constraint that is totally opaque at this point (we will discuss it later). Step II indicates how to sequence the elements. As an example, an element ending with a tail code $= 1/2$ can be concatenated only with elements having 1 or 2 as head code. Step III specifies a constraint on sequencing, as only elements coming from different groups may be concatenated. Step IV is a grouping operation on the resulting sequence. Here "metrical units" come into play, as the final chain is subdivided into verses made up of 4 metrical units (again, the number 10 is mysterious, more on this later). The final poem has to be a sort of sestina (Brancaleoni, 2007), as it is made up of six stanzas of six lines. In order to study the system as proposed by the "instructions", we first implemented a program based on a graph model, as instructions define each element as having two predecessor and two successors in the set of the 15 elements.

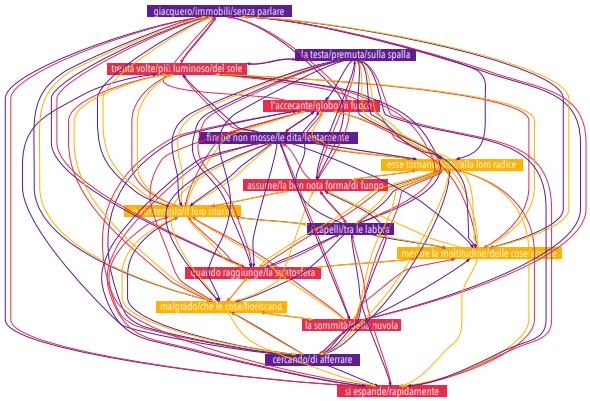

Figure 2: *Tape Mark I* graph.

Figure 2 shows a plotting by the Graphviz package of the graph that results from the data structure of predecessors/successors. Vertices represent elements, their color indicate the group they belong to, edges link to successors, their color being related to the predecessor. The graph is a direct, cyclic graph, and while not totally connected, it still does not reveal a specific topology (e.g. it is not a power law

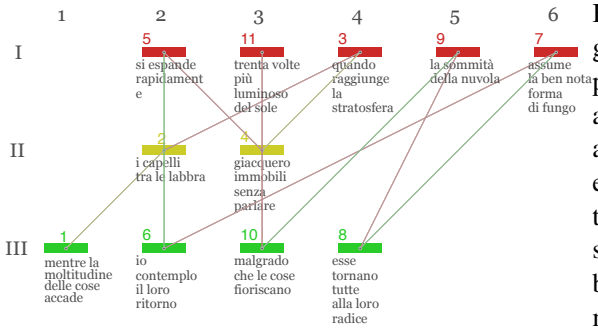

Figure 3: *Tape Mark I* as a path on a graph.

graph, where some vertices are densely connected). Such a topology seems to suggest a sort of uniform distribution of the elements, that more or less share the same rank. The graph represents all the virtual sequencing possibilities of the system as it takes into account also the Step II constraint on adjacent elements from different groups. An automatic visualization (using the Python-based Nodebox package) of a possible poem as a path on the graph is shown in Figure 3, where the vertical axis represents groups (the group is an attribute of vertices), the horizontal one represents elements, and each vertex is labeled on top with the sequence index (starting from 1 in group III, element 3).

Such a modelization by means of a graph defines a possible syntax, and explicitly aims at introducing a generative perspective, where each path on the graph is a possible poem. Being the graph cyclic, theoretically a path can be infinite, and the same vertices may be traversed more times. Here Step I comes into play, as it states (but this can be assessed only taking into account the implementation, as we will see) that no repetitions are possible. So Figure 3 shows a path with no repetitions, as vertices –once traversed– are no more available. This constraint results in various valid paths for a maximum of 15 vertices, the shortest valid paths having length $= 8$. Indeed, the graph model reported in Figure 2 can be thought as a grammar, as it falls in the set of the regular grammars in the Chomsky hierarchy, i.e. the simplest form of generative grammar, which is equivalent, in the recognition process, to a finite state automaton (Hopcroft et al., 2006). Such a simple grammar modelization poses an interesting question: was Balestrini in need of a computer? By drawing a graph on a paper sheet, the handmade generation of sequences is not a big deal. The key point is that Balestrini is not thinking in terms of a grammar-based model[1]. So, even if Balestrini's algorithm as provided by the "instructions" can be easily modeled as a generative procedure, it was not thought in these terms. Rather than in a generative, syntactic fashion, Balestrini was thinking in a combinatorial one. The graph model indicates that the maximum length of a sequence is 15. But the final poem is much longer (six stanzas of six verses). This can be understood only by inspecting the low level algorithm, that explains instruction I.

4 *Tape Mark I*: low-level model

Apart from the "instructions" section, the description of *Tape Mark I* reported in the *Almanacco Bompiani* (Morando, 1962) contains a section called *Elaborazione del Calcolatore* [computer processing]. It shows a complex flowchart (see Figure 4) that was probably completely opaque to the readers and intended only to document the "esoteric" low-level machine level, while the plain language "instructions" were its "exoteric", high-level side. Nevertheless, the flowchart allowed us for a more precise reconstruction of the original system[2].

The original program for *Tape Mark I* was written in the IBM 7070 assembler called AUTOCODER (IBM, 1961). From the description and by inspecting the flowchart of the low-level algorithm, we can reconstruct the memory organization adopted by the programmer (the engineer Alberto Nobis). The memory was organized in four tables, called Table-A, Table-B, Table-C and, not mentioned in the text, Table-(d) (see Figure 5). Table-A contains the original text fragments, i.e. each cell contains an element $(1 \rightarrow 15)$. A notable consequence is that each cell has a variable length. Table-B contains pointers, i.e. each cell contains two pointers to the beginning and to the end positions in Table-A (in Table-B, Bn and En respectively indicating BEGIN and END), of a specific element. Table-C contains four

[1] It might be noted that Chomsky's *Syntactic structures* was published only four years before (1957). Nevertheless, Balestrini (Morando, 1962) explicitly mentions grammars as a future reference for his work.

[2] Other information on the technical aspects, probably from an interview to Nobis, are reported by (Comai, 1985).

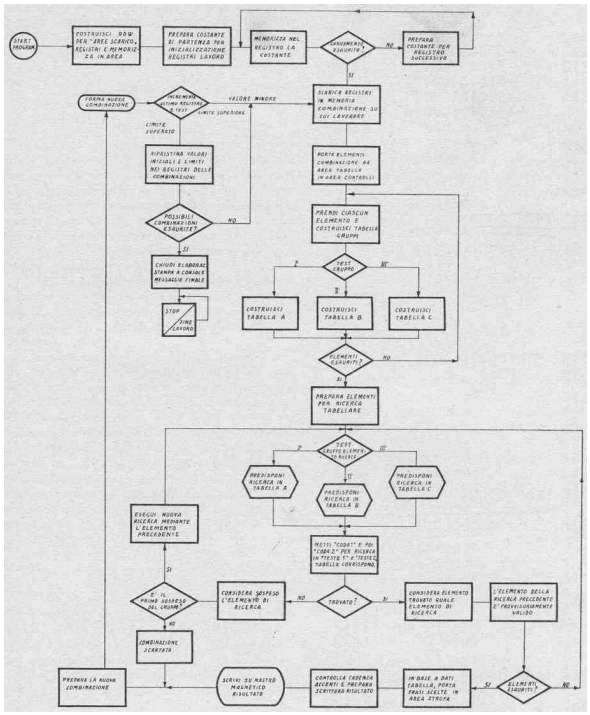

Figure 4: Flowchart by A. Nobis showing the memory procedures for table filling and result testing in the AUTOCODER implementation.

distinct data: (i) the head code of a specific element (HC); (ii) the tail code of the element (TC); (iii) the group to which the fragment belongs to (Gr); (iv) the position in Table-B of the pointer (BnEn). Finally, Table-(d) contains the combination of the positions of the Table-C, which correspond to the combination that has been initially extracted from a permutation of 10 over 15 elements. That is, Table-(d) contains a serie of indexes $[1, \ldots 15]$ that represents the elements to be permuted at the initialization phase of each cycle.

By this organization of the memory we understand that the low-level algorithm essentially works on pointers, i.e. the permutation of the fragments essentially consists of a permutation of memory positions of Table-C, and each possible combination extracted from this permutation is essentially a sequence of 10 positions (pointers) of the Table-C.

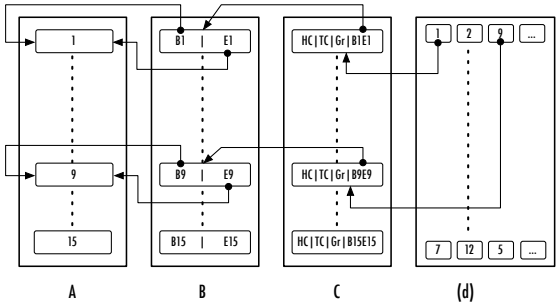

Figure 5: Memory organization of *Tape Mark I* implemented algorithm.

Here the number 10 comes into play, the one that was mysteriously mentioned by Balestrini in the "instructions". It is not clear why Balestrini introduces this constraint, and both aesthetic (high-level) and technical (low-level, due to pressing memory constraints on the IBM 7070) explanations are possible. This means that the whole process at each run takes into account (and generates) a $10-$element sequence. This memory organization probably also explains why elements are always provided with two head and two tail codes. The latter feature may be seen as a feedback constraint from low- to high-level (i.e. the "instructions"). To have a variable number of head and tail codes would have meant to define a further pointer table to take into account their variable length.

5 The simulation experiment

The flowchart in Figure 6 is the conceptual schema of *Tape Mark I* and formalizes the steps performed both by the human ("author") and by the machine. In step I ("Generation"), the algorithm starts from

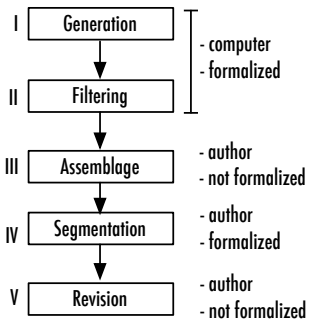

Figure 6: The conceptual schema of the *Tape Mark I* artwork

a permutation of the 15 fragments (provided by the programmer) and computes a specific combination. This means that the algorithm follows a brute-force approach, without any notion of valid sequence. In step II ("Filtering"), the computer removes the combinations that do not respect the rules expressed by the high-level algorithm. Step III is dedicated to the assemblage of valid sequences. The previously reconstructed methodology, strongly constrained by memory allocation techniques on the IBM 7070 and by AUTOCODER specifications, makes clear that the whole poem cannot be generated in one single run of the program, as the single run outputs a 10-element sequence. Thus, the poem is an assemblage of various outputs, an operation executed by Balestrini, that selects a number of combinations to be used together to produce the final opera[3]. In step IV ("Segmentation"), the author segments the combination in order to respect the chosen metrical constraints, i.e. 6 stanzas of 6 verses; In step V ("Revision"), the author adjusts a number of words in order to satisfy morphosyntactic constraints in the final text (e.g. verb-subject and number agreement).

One of the main goals of this paper is to understand the effort of the author, in other words what is the contribution of the poet in the *Tape Mark I* "electronic poem" (as computer-based poetry was called at times).

The most unclear point in the Balestrini's work is step I. Indeed, this step consists of two subprocesses: I-a) generate one permutation P of the 15 elements among the 15! possible permutations (1.307.674.368.000); I-b) generate all the possible combinations of 10 elements from P (i.e. without repetitions and permutations (Mazur, 2010)): for each P there are 3003 ($C(15:10)$ where C is the binomial coefficient) possible combinations[4].

The total number of possible outputs of *Tape Mark I*'s step I is huge: $P(15) * C(15:10) = 3,926,946,127,104,000$. However, many of these sequences are identical, since the number of distinct sequences is $P(10) * C(15:10) = 10,897,286,400$. Finally, the total number of "valid" sequences, i.e. sequences respecting the constraints of instructions II and III, that we computed by generation and test, is $65,284,636$.

One of the goals of the simulation is to understand how often the *Tape Mark I* was able to produce a valid sequence of fragments in output. So, we have implemented a program which reproduces the steps I and II of the flowchart in Figure 6. The original *Tape Mark I* program was able to generate the 3003 possible combinations of a single permutation in 660 seconds on the IBM 7070. We have implemented an optimized version of the same process by using C++: this program runs in 0.01 seconds on a modern laptop (4GB ram, i7 2GHz processor) to generate and test the 3003 possible combinations of a single permutation.[5] However, also with this fast program, we would need 414 years to test all the possible 15! permutations. So we decided to perform an experiment on ten millions random permutations of the 15 elements: for each permutation, we counted how many of the 3003 combinations were valid, i.e. how many combinations satisfy the constraints expressed in the high-level model. In this way, we can figure how often the original program produced an output that the poet could modify in the steps III and IV and V of Figure 6.

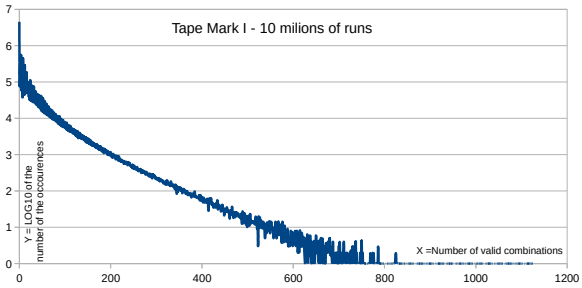

Figure 7: The number of occurrences of valid combinations found in 10 millions random permutations of the 15 elements.

We found that statistically half of the times all the 3003 combinations extracted from a permutation do not satisfy the required constraints. However, we also found that the maximum number of valid combinations from one single permutation was 1126 (see the logarithmic graph in Figure 7). So, this simula-

[3]This is evident in the poem by comparing the final verse of first stanza with the initial verse of the second one. They both belong to the same group (II), so the non-adjacent constraint of instruction II does not apply, as they are generated from two runs.

[4](Balestrini, 1962) incorrectly reports 3002.

[5]A first, non optimized version was implemented in Clojure and required 0.1 seconds.

tion confirms that in order to produce *Tape Mark I* the poet needed to run the program several times, adopting a severe trial and test procedure. Figure 8 shows an excerpt of the raw output of the system once printed on paper[6]. Three outputs of 10 elements are shown, the one on top is annotated to allow the reader to follow the chaining mechanism based on head and tail codes.

6 Balestrini vs computational creativity

In the light of the archaeological focus of our contribution we will not directly question the notion of computational creativity, rather we will discuss some aspects of Balestrini's work in relation to his computational practice and its critical reception. That is, we will address the question: what is (*ante litteram*) computational creativity for Balestrini in the Neo-avantgarde context of the '60s? In the following we identify some key features.

Formalization and metalanguage: a metalinguistic tension is a defining element in Balestrini's work, as content is subordinate to the explicit operations at the basis of its generation. The second poem collection by Balestrini is titled *Come si agisce* ("How to act") and its final section is a table that precisely specifies how the poems in the collection were created, and thus how to possibly create other, new poems: thus, "poetry is an operation, the poet shows, precisely, how to act" (Brancaleoni, 2007, 125). Not by chance, *Tape Mark I* has been overtly described in the *Almanacco*. To formalize the poetic operation -as noted by Sanguineti, poet but also prominent critic of the avantgarde- the use of the computer is in some sense a natural consequence of such an aesthetics: "electronic poetry is [...] the natural extreme outcome" of a similar aesthetics (Sanguineti, 1965, 72)[7]; **Materiality of language:** language, primarily on its expressive surface, and secondarily in relation to the conveyed content, is the matter of poetry. The poetry is intended on the one hand to demystify language by suspending its actual practicality, on the other hand to drastically destroy meaning, in order to reach Barthes' "Degree Zero" of language, the level of its materiality (Brancaleoni, 2007). Computers allow to directly target this goal by efficiently providing symbol manipulation;

Redefinition of the role of the reader: thanks to digital printing, in the new Italian and English editions of Balestrini's novel *Tristano* (2015-16) (Balestrini, 2015; Balestrini, 2016), each copy is different from any other, thus reaching his original purpose, i.e. to escape "the rigid determinism of the mechanical Gutenberg printing process" (Balestrini, 2015). In the preface of the novel, Eco has individuated three radically different "roles of the reader" (Eco, 1984) implied in such a literary device, that indeed are at stake also in the case of *Tape Mark I*.

1. pick up a copy and read it as if it were original and unchangeable;
2. find multiple copies and retrace the different outcomes of combinatorics;
3. choose one among the many texts on the basis of the reader's evaluation criteria (Balestrini, 2015).

Such a dispersion of roles is possible only in case of usage of computer-controlled generative processes;

System vs text as a value for the open work: while poetry is typically placed at the text level, in Balestrini's approach it is the (generative) system at its origin that is considered in itself as a value, as the project is to undermine the dogma of the original, unique and definitive literary work. As noted by Eco,"the whole work resides in its variations, even in its variability. The electronic brain has made an attempt to create an *open work*" (Eco, 1962, 185)[8]. Hence the relevance of permutation, made possible only by computational means, as an exhaustive deployment of all the possible outcomes. Another historical example of such a permutative fury is Queneau's *Cent mille milliards de poèmes*, that was published exactly in the same year of Balestrini's *Tape Mark I*. Queneau, one of the founders of Oulipo, devised a typographical setting in which each sheet was cut into stripes. By turning the stripes, new poems emerge from the combinations of various lay-

[6]Initially, generated data were stored on magnetic tape, hence the name of the work. The final print was on a 63.74 meter continuous roll (Morando, 1962).

[7]Interestingly, (Colton et al., 2014) argue the relevance for the user to meta-linguistically document computational creative processes.

[8]Here Eco is referring to his notion of "open work" as a system of interpretative possibilities, that was originally published exactly in 1962 (Eco, 1989).

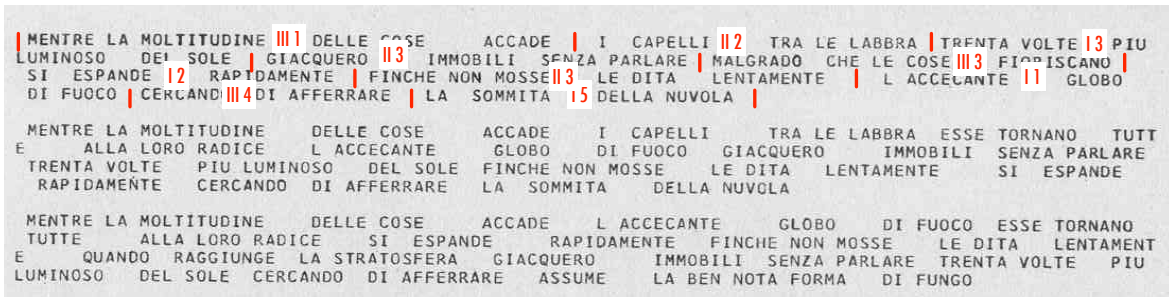

Figure 8: An excerpt of the raw output (four 10 element runs). On top, an annotated sequence showing the group and relative element index.

ers. It is interesting to note that Queneau aimed at creating "une sorte de machine à fabriquer des poèmes" (Queneau, 1961).

Poet as a distributed demiurge: as noted by Sanguineti, "the divine fury of the poet [...] is converted into the infinite technical possibilities of the electronic instrument, elected both as the imaginative stimulus and as the practical manufacturer" (Sanguineti, 1965, 75). Hence the subject gains a role of mediator, and it is distributed at various levels, in a shared association with machine's symbolic agency. Subjectivity thus emerges:

- in the choice of materials, both in terms of the source texts and their cutup;
- at the syntactic level (the definition of the grammar, even if the term does not properly apply);
- in the selection and assemblage of the outputs and in the final revision (punctuation and syntactic agreement)[9].

To sum up, it is worth emphasizing again Sanguineti's observation: computational poetry is the natural extreme outcome of such an aesthetics. In the case of *Tape Mark I* creativity is intrinsically computational as it is intrinsically shared between man and the machine.

7 Conclusions

Our aim was to inspect into details a substantially well documented example of computer-generated poetry. A first interesting result is that low-level features, that depends on available technology at a certain historical time, have a crucial impact on the output, as evident in the friction, so to say, between high- and low-level algorithms. Is *Tape Mark I* a good example of computer-generated poem? The answer to this question is simply yes, as *Tape Mark I*, far from being an experiment, is a crucial case, highly considered in literature (hence, its interest). And, thus, can the operations at its basis be considered relevant for a general model of computer poetry? The procedures devised by Balestrini are intrinsically local to his aesthetic vision and to the historical context (including the technological one, as we discussed). In this sense, our study seems to suggest that, differently from natural language, the results of poetry (and of all aesthetic objects) must be assessed in relation to its *Wirkungsgeschichte* (Gadamer, 2004), that is, the history of its effects on a certain community.

[9] In any case, Balestrini was positive on a future automatization of the Revision step (Morando, 1962).

References

Oscar Alicicco, Laura Mastroddi, and Federica Romanò, editors. 2010. *I novissimi. Ricostruzione del fenomeno editoriale*. Oblique Studio, Roma.

Nanni Balestrini. 1962. Tape Mark I. In Sergio Morando, editor, *Almanacco letterario Bompiani 1962: le applicazioni dei calcolatori elettronici alle scienze morali e alla letteratura*. Bompiani.

Nanni Balestrini. 1968. Tape Mark I. In J. Reichardt, editor, *Cybernetic Serendipity: The Computer and the Arts : a Studio International Special Issue*, Studio international. Special issue. Studio International.

Nanni Balestrini. 2015. *Tristano*. DeriveApprodi, Roma.

Nanni Balestrini. 2016. *Tristano*. Verso, London –New York.

Margaret A. Boden. 2015. Foreword: How Computational Creativity Began. In Tarek R. Besold,

Marco Schorlemmer, and Alan Smaill, editors, *Computational Creativity Research: Towards Creative Machines*, volume 7. Atlantis Press.

Claudio Brancaleoni, 2007. *Re-lab: immagini parole*, chapter "La rivoluzione in forma di parole": Nanni Balestrini, pages 125–136. Morlacchi, Perugia.

Simon Colton, Michael Cook, Rose Hepworth, and Alison Pease. 2014. On acid drops and teardrops: Observer issues in computational creativity. In *Proceedings of the 50th Anniversary Convention of the AISB*. Society for the Study of Artificial Intelligence and Simulation of Behaviour.

Adriano Comai. 1985. Poesie elettroniche. L'esempio di Balestrini. Master's thesis, Università di Torino, Torino.

Umberto Eco. 1962. La forma del disordine. In Sergio Morando, editor, *Almanacco letterario Bompiani 1962: le applicazioni dei calcolatori elettronici alle scienze morali e alla letteratura*, pages 175–188. Bompiani.

Umberto Eco. 1984. *The Role of the Reader*. Indiana UP, Bloomington.

Umberto Eco. 1989. *The Open Work*. Harvard UP, Cambridge, Mass.

Christopher T. Funkhouser and Sandy Baldwin. 2007. *Prehistoric Digital Poetry: An Archaeology of Forms, 1959-1995*. Modern & Contemporary Poetics. University of Alabama Press.

Hans-Georg Gadamer. 2004. *Truth and Method*. Continuum, New York.

Gervás, Pablo. 2015. Deconstructing Computer Poets: Making Selected Processes Available as Services. *Computational Intelligence*.

Pablo Gervas. 2016. Constrained creation of poetic forms during theme-driven exploration of a domain defined by an n-gram model. *Connection Science*, 28(2):111–130, April.

Alfredo Giuliani, editor. 1961. *I novissimi. Poesie per gli anni '60*. Rusconi e Paolazzi, Milano.

John E. Hopcroft, Rajeev Motwani, and Jeffrey D. Ullman. 2006. *Introduction to Automata Theory, Languages, and Computation (3rd Edition)*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

IBM. 1961. Reference manual ibm 7070 series programming systems autocoder. Technical report, IBM, New York.

Vincenzo Lombardo, Andrea Valle, Fabrizio Nunnari, Francesco Giordana, and Andrea Arghinenti. 2006. Archeology of multimedia. In *Proceedings of the 14th ACM International Conference on Multimedia*, MM '06, pages 269–278, New York, NY, USA. ACM.

Brent MacGregor. 2002. Cybernetic serendipity revisited. In *C&C '02: Proceedings of the 4th Conference on Creativity & Cognition*, pages 11–13, New York, NY, USA. ACM.

D.R. Mazur. 2010. *Combinatorics: A Guided Tour*. MAA textbooks. Mathematical Association of America.

Sergio Morando, editor. 1962. *Almanacco letterario Bompiani 1962: le applicazioni dei calcolatori elettronici alle scienze morali e alla letteratura*. Bompiani.

Raymond Queneau. 1961. *Cent mille milliards de poèmes*. Gallimard, Paris.

Jasia Reichardt, editor. 1968. *Cybernetic Serendipity. The computer and the arts*. Number Special Issue. Studio International, New York.

Ehud Reiter. 2007. An architecture for data-to-text systems. In *Proceedings of the Eleventh European Workshop on Natural Language Generation*, ENLG '07, pages 97–104, Stroudsburg, PA, USA. Association for Computational Linguistics.

Gian Paolo Renello. 2010. *Machinae. Studi sulla poetica di Nanni Balestrini,*. Il Castello di Atlante. Bologna, CLUEB.

Edoardo Sanguineti. 1965. *Ideologia e linguaggio*. Feltrinelli, Milano.

Jukka M. Toivanen, Hannu Toivonen, Alessandro Valitutti, and Oskar Gross. 2012. Corpus-based generation of content and form in poetry. In *Proceedings of the Third International Conference on Computational Creativity*, page 211–215, Dublin, Ireland, may.