Artificial Creativity: combining Evolutionary Computing with LLMs for creative scriptwriting

Rodrigo Silva Dias Lopes
Instituto Superior Técnico
Lisbon, Portugal
rodrigo.d.lopes@tecnico.ulisboa.pt

***Abstract*—This thesis explores the integration of Large Language Models (LLMs) with Evolutionary Computing, as a tool for creative writing, particularly in the domain of movie idea generation. Our work adopts an exploratory approach, designed to assist human scriptwriters (users) in their creative process.**

Our methodology involves generating new movie plots based on traits of existing films, using them as 'inspirational' movies, along with user-provided guidelines. In this context, movies are treated as free-form text objects, that will be generated and manipulated by the LLM and evolved by the Evolutionary Algorithm. The usage of a Genetic Algorithm (GA) serves the purpose of combining, mutating, selecting and evaluating new ideas for movie plots, allowing the most-promising ideas to be further exploited. The LLM will be employed during the recombination, mutation and evaluation phases of the GA. Since the evaluation phase involves utilizing different objective functions to determine the overall quality of the newly generated ideas, both single- and multi-objective optimization methods were tested, alongside GPT-3.5 and GPT-4o models.

Our approach showed significant potential, with movie plots evolving through generations, resulting in new ideas that meet the desired requirements. We consistently generated new plots by selecting three 'inspirational' movies from a database of 42,306 films. Best results were achieved with GPT-4o, single-objective optimization and a large population size. Finally, we collected human feedback to validate the fitness scores assigned by the LLM. Regardless of the limitations encountered, our work successfully combined different Data Science paradigms to generate creative content.

***Index Terms*—Large Language Models; Evolutionary Computing; Genetic Algorithms; Evolution Through Large Models; Scriptwriting**

I. Introduction

The scope of this work is to construct a tool that helps people at performing creative tasks. In particular, this work will focus on one particular task: writing movie plots. The key idea that motivates our work is the potential to evolve ideas through an Evolutionary Algorithm (EA), allowing us to explore different ideas while privileging the most promising ones. Conceptually, writing a movie can be viewed as generating a piece of free-form text that encapsulates ideas and descriptions. This was convenient because it allowed for the integration of Large Language Models (LLMs) to handle and manipulate movie descriptions. The fact that these movie plots are seen as pieces of text means that the work presented in this thesis can be easily extended to any other creative task that can be put into text form.

As a tool for brainstorming new movie ideas, this work features two novel aspects. Firstly, the movies generated by the LLM should take "inspiration" from existing movies, similar to how a human writer would draw inspiration from existing narratives. Secondly, the user should have the possibility to provide an initial text input to the model to guide the creation process. This ensures that the artificially generated plots will absorb both the characteristics of the chosen existing movies and the user's creative input.

Furthermore, within the evolution process, the LLM has the additional task of providing feedback on the quality of the generated ideas. It assigns a fitness score to each generated plot, helping to identify which new ideas are more promising. This means that the evaluation process is pivotal in guiding the EA, but we cannot know *a priori* if the LLM is doing a fair evaluation. To assess the quality of the evaluation made by the LLM we compared it with evaluations given by humans, measuring the alignment between both.

In conclusion, this work features two main goals. The primary goal had an exploratory nature: we want to investigate the possibility of evolving creative ideas within an evolutionary framework. Specifically, we aimed at integrating LLMs with a Genetic Algorithm (GA) to evolve a population of movie ideas and assess whether there is any kind of improvement across generations. The LLM has the roles of combining, mutating and evaluating movie plots. The second goal, closely related to the first, was to ensure that the outcome of this new framework can effectively assist scriptwriters in their brainstorming and creative process.

II. Related Work

Lehman et al. (2022) [1] introduced Evolution through Large Models (ELM), which is a framework for Evolutionary search for code that uses LLM as a tool to generate intelligent code mutations. In this paper, the authors sustain that Evolutionary Computing and Deep Learning (particularly, Generative AI) are not competing paradigms, but instead complementary. Bradley et al. [2] introduced a new approach known as Quality-Diversity through AI Feedback (QDAIF), which is an extension of ELM, where candidate solutions undergo the LMX mutation, a crossover operation performed by an LLM proposed by [3]. Lanzi et al. [4] developed an interactive evolutionary framework for collaborative game design, where the EA variation operations are performed by an

LLM. Together with [5] and [6], all the papers aforementioned combine the Evolutionary framework with LLMs.

AI Feedback refers to when one LLM provides feedback on the quality of the output generated by another language model. Madaan et al. further developed this concept in their work [7], where the feedback given by the LLM about the previously generated output is passed again to the LLM, further refining itself. QDAIF [2] and OpenELM [8] (a library implementing of ELM [1]) also utilize AI provided feedback in their works.

Our work is a unique combination of the techniques described in the cited articles – namely, it implements an integration of an LLM in a Genetic Algorithm (GA), where the evaluation operation is based on feedback provided by the LLM. In addition, our work focuses on utilizing these techniques to stimulate creativity and evolve ideas for movies, which presents, to the best of our knowledge, a novelty in the field.

III. Methodology

The fundamental idea underlying this work is that a group of movie descriptions will make up a population, upon which a GA will be applied together with an LLM. Each individual in the GA, representing a movie plot, is simply a piece of free-form text that describes in detail a certain movie, including information about its title, genre, main characters, world-building description and its summary. Being pieces of text, each of these individuals can be conveniently manipulated by an LLM during the recombination, mutation and evaluation phases of the GA. This work employed the models GPT-3.5 and GPT-4o into its functioning.

Figure 1 illustrates the main components of the architecture, highlighting their connections. This section will explain each step of the GA, including the initialization, the operators (variation, selection and evaluation) and the optimization mode, ending with a description of the experiment done to validate the fitness function through human feedback.

A. Initialization of the GA

The Genetic Algorithm (GA) typically begins by initializing the population randomly, which in our specific context might involve, for example, asking an LLM to generate a set of random movie plots. However, this approach is not aligned with the core premise and goals of our work — that is, to combine a set of pre-existing movie plots with a user guideline by utilizing an LLM and iteratively evolving these ideas with the EA.

Instead of a random initialization, we asked the user to:

1) Select an arbitrary number of existing movies that will serve as inspiration for generating the new plots (e.g. ”The Matrix”, ”Back to the Future”, ...);
2) Optionally, provide some instructions to guide the generation of new plots (e.g. ”Write a dystopian movie set in the near future.”).

The process begins by generating a set of movies that agree with the given guidelines, by passing to the LLM the prompt: ”Write a movie plot from scratch, following these instructions:”. User guidelines are concatenated to this prompt. This process repeats S times, generating S new movies based on the user's instructions.

Then, considering a population of N individuals, the remaining $N - S$ movies are created by combining and/or mutating the set of movies chosen by the user, once again, by employing an LLM. The summaries of the movies previously chosen are retrieved from a corpus of 42,306 movies developed by [9]. This process of combining/mutating will be explained in section III-B.

Figure 2 presents an illustrative diagram about the initialization process. In short, initializing the EA in this way ensures the incorporation of the two desired key elements into the population — ”inspiration” drawn from the chosen movies and from the user's instructions.

B. GA operators

In the context of this work, an LLM can be seen as a block that receives a piece of text — a prompt — and outputs another piece of text, that obeys the prompt. This is the key mechanism for the recombination, mutation and evaluation operators. By applying recombination and mutation to movie ideas, we can explore new creative concepts. If an idea is promising (i.e., assigned a high fitness value), it is more likely to be selected for further reproduction, thus propagating the potential of that idea into the next generations.

Recombination Operator

The LLM is prompted with the task of combining two individuals A and B (movie descriptions), creating one new individual — that should be an original plot, taking ”inspiration” from both parents — by passing the prompt: ”Create a new movie plot combining the two provided plots A and B, by merging their features.”

Mutation Operator

Similarly, the LLM is prompted with the task of modifying one individual, following a specific mutation — for example, the mutation could be something such as ”Introduce unexpected plot twists or revelations that change the direction of the story”. The list of possible mutations includes 24 different mutations like the one cited.

Selection Operator

The Selection operator is responsible for choosing the fittest individuals from the current population to pass their genes onto the next generation. It plays a crucial role in guiding the evolution process by ensuring that higher-quality solutions have a better chance of being selected for reproduction.

We employed two techniques:

1) Elitism, meaning that the best M individuals (highest fitness), at each generation, pass directly on to the next generation, without suffering any change;
2) Tournament selection with size $q = 2$, meaning that when selecting a new parent, two individuals will be randomly drawn from the population, but only the most-fitted will have a chance to reproduce and pass its information onto the next generation.

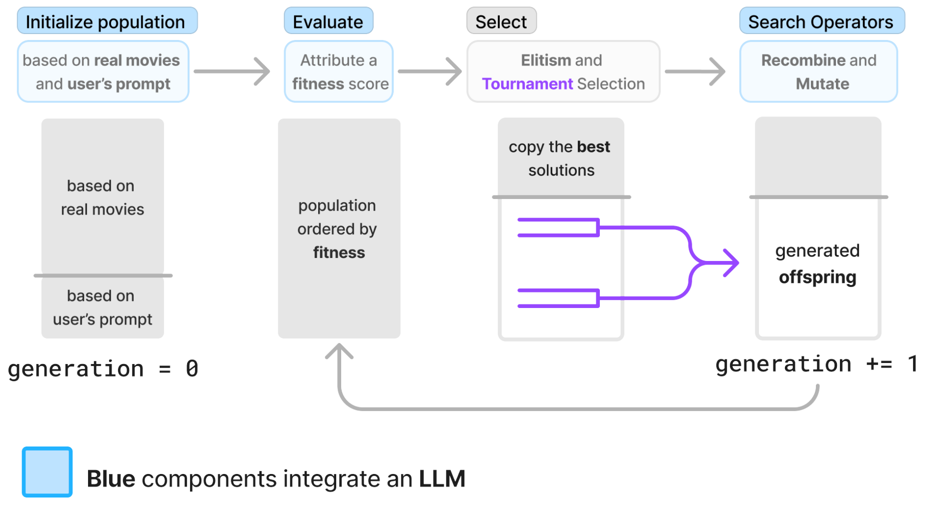

Fig. 1. Architecture Overview

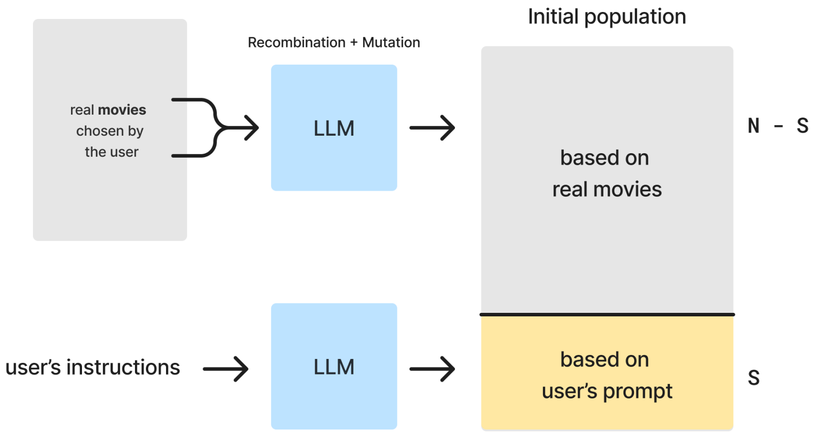

Fig. 2. Initialization schema

Evaluation Operator

The selection operation is based on a fitness value that is attributed to each individual — naturally, more fitted individuals are more likely to generate offspring, while the ones with lower scores will be most likely discarded. The fitness function is composed of 4 different objective functions to be maximized, 3 of which involve asking an LLM to evaluate a certain aspect of a movie. The fitness objectives include:

1) **Number of Ancestors**: counts from the set of movies initially chosen by the user, how many are ancestors of movie X. We divide this number by the total number of selected movies, and scale the result from 0 to 10. We want to maximize this metric since it means that movie X descended from *more* movies from the set of initially chosen movies. In other words, it carries more information from those movies, which matches the goal of this work – creating new movies 'inspired' by existing ones.
2) **Quality**: Evaluates the overall quality of the movie. The LLM is asked to rate the movie plot on a scale of 0 to 10 across 15 different aspects, which include coherence, originality, consistency of characters' traits, and more. The set of quality criteria was thought to cover all possible dimensions inside a movie plot that can be evaluated. An average quality value is calculated, for each plot.
3) **Inspirations**: Measures how much the new movie captures influence or is inspired by the plots of the initially chosen movies. It works by passing to the LLM a prompt starting with: "I will give you two movie plots. You will evaluate from 0 to 10 how similar they are".
4) **Instructions**: evaluates how well the generated movie obeys the initial user's guidelines. The evaluation prompt includes: "I will give you Instructions in text form. You should transform them into bullet points. Then, I will give you a movie plot in text form. You will evaluate it

from 0 to 10 considering each one of the bullet points and give a brief justification."

Algorithm 1 recaps the evaluation process, which involves assigning to each new individual set of fitness values, each one corresponding to one objective function. Note that the prompts cited in this section consist only of a passage of the full prompt. The complete prompts were designed also to specify the desired output format, and incorporated prompting engineering techniques, such as chain-of-thought (CoT) or few-shot learning, that improve the quality of the output.

Algorithm 1 Assigning Fitness Scores

```
instructions ← user's guidelines
for ind in individuals do
    evaluate number of ancestors (ind);
    evaluate quality (ind);
    evaluate instructions (ind, instructions);
    for movie in chosen movies do
        evaluate inspiration (ind, movie);
    end for
    compute average (evaluate inspiration);
end for
```

C. Optimization mode

Our work tested both single- and multi-objective optimization to guide the GA evolution.

Single-Objective Optimization (SOO)

The SOO approach handles the fact that we have 4 objective functions by transforming a multi-objective optimization problem into a SOO one. To do so, the 4 objective functions are combined into a single fitness function after applying a weighted sum. The weights were set empirically, where the '*Inspirations*' and '*Instructions*' fitness functions received a larger weight ($w = 3$) to emphasize the fact that the generated movies should contain significant 'inspirations' from the original movies and, at the same time, should respect the user's guidelines. On the other hand, we observed that the LLM was not good at attributing the '*Quality*' score, so we diminished its importance in the weighted sum ($w = 1$). The '*Number of Ancestors*' function got an intermediate weight ($w = 2$). The SOO algorithm runs until a pre-determined number of consecutive generations happen without improvement.

Multi-Objective Optimization (MOO)

The classical way of dealing with more than one objective function is by tackling the problem as a MOO problem. To do so, we employed the NSGA-II algorithm, which sorts the population by Pareto fronts and, inside each front, by crowding distance in descending order. Therefore, once we have sorted the individuals, selection can be applied, prioritizing the reproduction of highly fitted individuals.

D. Validating the Fitness Function

Defining an accurate fitness function is crucial when it comes to controlling the evolutionary process: a good evaluation will correctly guide the evolution towards an optimal solution, pressuring the 'good' genes to spread over the generations. Since, in our work, we are performing evaluation through an LLM, the following question arises: how can we be confident that the evaluation is accurate?

Given the inherent subjectivity of this task, we cannot be certain that the fitness scores assigned by the LLM are fair. Human evaluation, while potentially more reliable, would quickly become expensive and time-consuming. However, we can assess the validity of the LLM-powered fitness function by taking *a few* examples of LLM evaluations, asking people to perform the same evaluation and checking if the two evaluations align.

Because we are interested in verifying if the human and LLM preferences agree with each other, there is no point in asking people to rate a movie from 0 to 10. Instead, it was asked in the survey to order by preference (best to worst) a set of four movies, for each one of the objective functions. This makes a ranking (e.g. $A \prec D \prec C \prec B$).

With this in mind, we conducted a survey containing 4 movie plots generated by the LLM. Participants were asked to rate the movies from 1 to 4, and we compared their rankings with the evaluations given by the LLM. It is worth noting that the LLM doesn't produce directly rankings, instead, it assigns scores from 0 to 10. However, once the scores are attributed to the set of 4 movies, it is straightforward to convert them into a ranking, with the highest-scored receiving a rank of 4, the second-highest a rank of 3, and so on.

In total, we designed 6 surveys, each one representing a different scenario, *i.e.*, the plots inside each one were subject to different initial configurations (distinct guidelines and movies chosen by the user). Each survey, corresponding to one of these scenarios, asked human evaluators to rank 4 plots in the exact same aspects evaluated by the LLM. Specifically, we asked humans to decide on:

1) *Quality* — which movies were best overall;
2) *Inspirations* — which plots were more similar to the parent movies;
3) *Instructions* — which movies better followed the given instructions.

Table I presents the initial setup for the generation of new movies in 2 of the surveys, as an example. Each survey included new movies inspired by 3 existing films ('inspirational' movies) and followed our guidelines (instructions). We made 6 different surveys, each containing 4 movies generated by our algorithm.

After collecting human and LLM feedback, the agreement between the two rankings can be measured with Spearman's ρ coefficient [10], given by:

$$\rho = 1 - \frac{6\sum d_i^2}{n(n^2-1)} \tag{1}$$

where d_i is the difference between the ranks of each observation, and n is the ranking size. It measures the correlation between two rankings, where 1 indicates perfect correlation, -1 means negative correlation and 0 indicates no correlation.

TABLE I
EXAMPLE OF TWO INITIAL SETUPS FOR THE MOVIES GENERATED IN TWO DIFFERENT SURVEYS. WE DESIGNED 6 OF THESE SURVEYS.

Inspiration Movies	**User Guidelines**
Back to the Future + Toy Story + Mamma Mia!	"*A young girl finds her old toys can time-travel, leading her on an adventure to meet her young parents and uncover family secrets.*"
The Incredibles + Aladdin + Batman	"*Five friends discover a magical lamp with a genie that grants superpowers and must stop an evil organization from using it for global domination.*"

TABLE II
EXAMPLE (EXCERPT) OF ONE MOVIE PRODUCED BY OUR ALGORITHM. *Jurassic Park*, *King Kong* AND *Godzilla* WERE CHOSEN AS THE INPUT ('INSPIRATIONAL') MOVIES, FOR THE GENERATION OF THIS PLOT. IT WAS PRODUCED USING GPT-4O FOR THE GENERATION (I.E., DURING RECOMBINATION AND MUTATION), AND GPT-3.5 FOR THE EVALUATION PROCESS.

Excerpt of "Island of Shadows"
Title: "Isle of Shadows" **Genre**: Adventure / Thriller **World-Building:** A permanent cloud bank hides an island inhabited by ancient dinosaurs and a giant ape worshipped by locals, while a past nuclear event in the region created an enormous sea monster. **Plot**: (...) - Scene 5: The tribe kidnaps Laura and offers her as a sacrifice to Kuro, a monumental ape. Kuro takes Laura into the jungle. - Scene 6: Grant and the team set out to rescue Laura. They encounter various dinosaurs and other dangers. - Scene 7: The team finds a sick Triceratops and helps it, showing the island's mix of creatures. - Scene 8: Marcus, a programmer, deactivates the island's security system to steal dinosaur embryos, releasing a T-rex. - Scene 9: The T-Rex attacks the team, causing chaos. Grant and Laura escape, but many team members are lost. (...) **Main Characters:** (...)

Since we had several human evaluators, it was necessary to aggregate their feedback into a single consensus ranking to compare with the LLM's evaluations. Feedback from different people, each providing their own ranking of four movies, was combined into a single consensus ranking using majority voting. Majority voting was performed employing Borda Count — the top-ranked movie in each ranking gets 4 points, the second gets 3 points, and so on until the last gets 1 point. The Borda Count is computed by summing up the points across all individuals' rankings. This summation of rankings is then converted back into a ranking itself (consensus ranking), which means that movies are ranked again from 1 to 4 — **Figure 3** illustrates this process.

In addition, we also decided to investigate the level of agreement among the individual human evaluators. To do so, we computed Spearman's ρ between every pair of human rankings and averaged them. The idea is that the resulting number captures the inter-human agreement, in an interval from -1 to 1. **Figure 4** illustrates the calculation of the inter-human agreement.

IV. RESULTS & DISCUSSION

Overall, the result of our work is positive: we were able to run the Genetic Algorithm (GA) paired with an LLM and we obtained plots of new and original movies after a few generations, with the majority of the movies respecting the requirements (this is, drawing inspiration from the chosen movies and respecting the guidelines provided by the user). Here, we shall discuss and analyze the experiments and results obtained with our algorithm.

A. *Qualitative analysis*

To understand the quality of the generated movie plots, let us take as the example "Isle of Shadows", an artificially generated plot inspired by *Jurassic Park*, *King Kong*, and *Godzilla*. For the generation, we provided the following guidelines: "*A team of military specialists and scientists must stop a giant ape and engineered dinosaurs, unleashed by an illegal genetics lab on a remote island, from reaching and destroying urban areas.*" An excerpt of the plot is presented here in **Table II**.

At first glance, the example in Table II looks like a nice and well-structured movie plot, and, in fact, it respects the desired requirements – it draws clear inspiration from the three input movies and follows quite well the provided guidelines. This suggests that our algorithm is successfully maximizing the '*Number of Ancestors*', '*Inspirations*' and '*Instructions*' objective functions.

However, if we inspect more carefully, we find some problems of inconsistencies or lack of originality. In our concrete example, we can notice illogical reasoning between scenes 5 - 9, where initially Laura was kidnapped and offered to Kuro (scene 5), and suddenly she appears together with Grant (scene 9), without any mention of how she was rescued in the meantime. In addition, some traces of the movie are clearly reminiscent of the original movies, raising the question of plagiarism. A few examples are: the scene where Kuro (a monumental ape) climbs the World Trade Center is highly reminiscent of King Kong's iconic climb up the Empire State Building; the character Dr. Samuel Grant shares many similarities with Dr. Alan Grant from Jurassic Park (both are

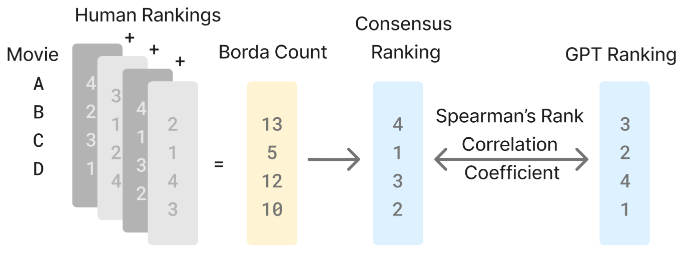

Fig. 3. Computing rank correlation between Human and GPT evaluations. Rank correlation is computed separately for each question in the survey.

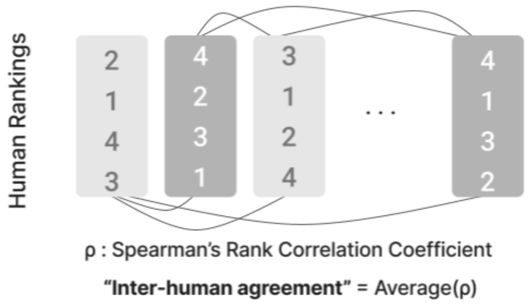
Human Rankings
ρ : Spearman's Rank Correlation Coefficient
"Inter-human agreement" = Average(ρ)

Fig. 4. Computing Inter-human Agreement. It is computed separately for each question in the survey.

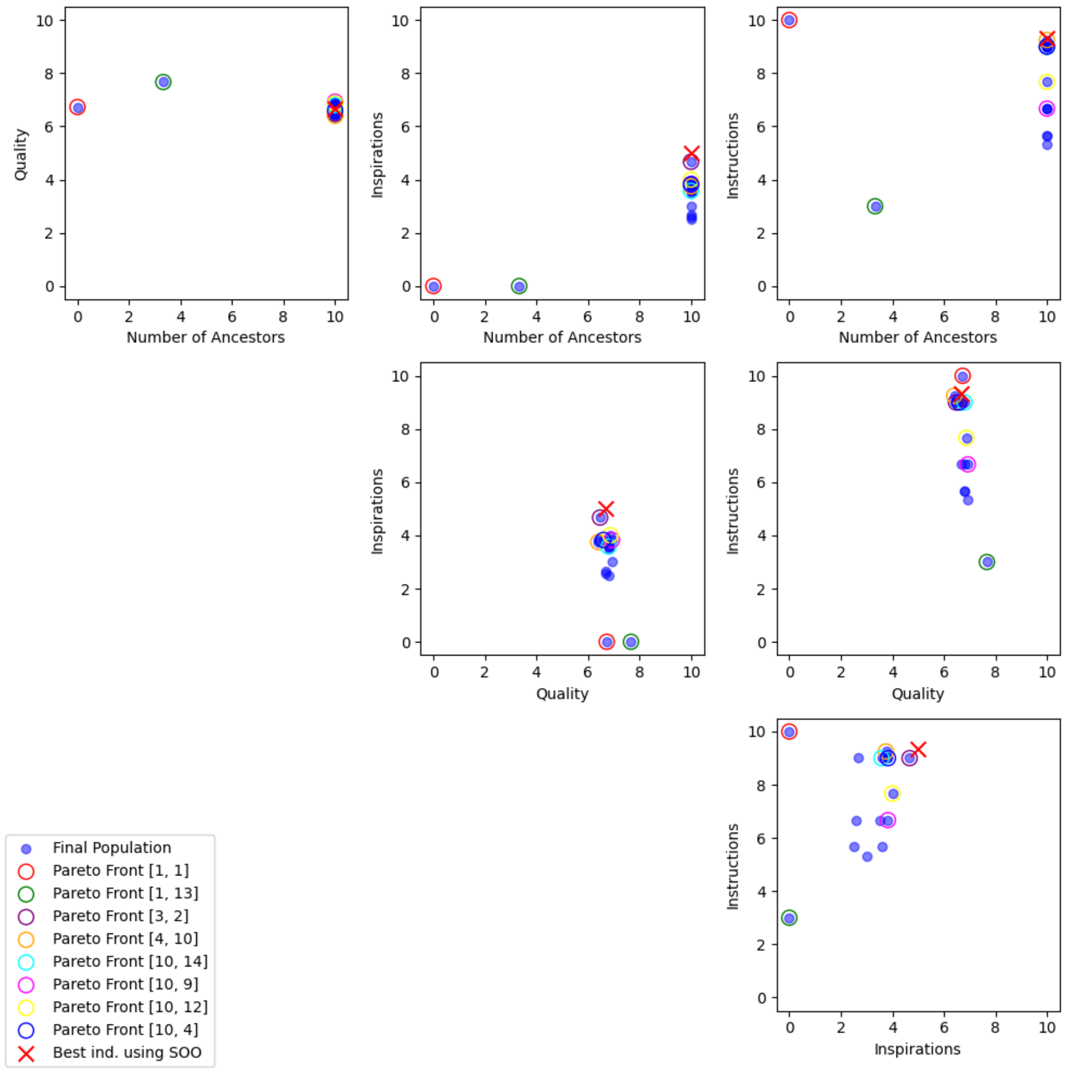

Fig. 5. Comparing SOO and MOO solutions in the objective space. Blue dots represent the final population generated with MOO, some of which have a colored circle around them, meaning that they are part of the first Pareto front. The red X represents the best individual generated with SOO for the same configuration.

paleontologists with the same surname); and the idea that Leviathan was created due to a nuclear incident ties into Godzilla's origins.

The fact is these types of phenomena appear in almost every generated movie. Our hypothesis is that this is due to two main reasons. Firstly, there is no reason to believe that the LLMs always reason logically, in fact, they suffer hallucinations often — producing illogical or incorrect statements. Secondly, the evolution process does not penalize individuals with this kind of "bad" characteristics, even though the '*Quality*' objective function was actually designed to look for both the coherence and originality of the given plot. The problem is that the '*Quality*' fitness function is performing poorly: the LLM has the tendency to attribute scores around 7 or 8, independently of the actual quality of the plot, meaning that it has lost its significance. This implies that these undesired characteristics still have the chance of propagating to future generations. We believe that the LLM is attributing 7 or 8 because it mimics human patterns, as people often rate vague or subjective aspects in this mid-range to indicate partial agreement.

In any case, it is safe to say that the implemented design worked, as the movie plots evolve through generations, resulting in new movie ideas that meet the desired user requirements.

B. *Discussing LLMs and Their Limitations*

The quality of our results is heavily dependent on the chosen language model. For instance, a robust, state-of-the-art model like GPT-4o is more likely to generate higher-quality plots and provide more accurate evaluations, whereas older or smaller models may not produce results that are as good.

In fact, GPT-3.5 often generates sentences that are too ambiguous, too general, without describing what truly has happened in the narrative. What we mean by this is that many times it does not reveal what really happens in the story. Instead, it uses expressions like "exposing the hidden secret", "dark secrets", "secrets are revealed", "unlocking the truth", "uncover the truth", "truth exposed", etc., without ever telling us what *actually* is the "secret" or "truth" in the context of that specific story. On the other hand, GPT-4o, despite not being perfect, is far more specific and detailed when writing a narrative.

In addition to the vagueness issue described above, GPT-3.5 writes, in general, lower-quality stories with respect to GPT-4o. One clear example of a limitation showed up when trying to combine movies: sometimes, instead of generating a new plot inspired in both parents, GPT-3.5 wrote two *parallel* narratives inside the same plot (the two narratives never crossed!).

To mitigate these undesired effects, other than changing the LLM to a better one, one can also apply prompting techniques to improve the response's quality. In fact, prompt engineering had a very satisfactory effect when tuning the LLMs to assign fitness scores, as no big difference was observed between GPT-3.5 and GPT-4o evaluations. However, prompting engineering only works to a certain extent and it may not resolve every limitation, namely when it comes to generating good and cohesive stories. In these cases, we may conclude from our study that utilizing a more powerful model — like GPT-4o — is much more effective.

C. *Analysis of the Evolution*

We consistently run our algorithm with diverse initial configurations (i.e., providing distinct sets of movies and guidelines), which allowed us to study the behavior of the GA evolution. We fixed the population size ($N = 15$), the maximum number of generations ($N_{gen} = 15$) and employed the GPT-4o model. Later on (Section IV-D), we will discuss the implications and potential advantages of increasing both N and N_{gen}.

Six experiments were carried on, with 3 input movies each, corresponding to the same initial configurations of the six surveys conducted, mentioned in III-D. For the analysis in this Section, let us take the two examples from **Table I**.

Figure 6 presents the evolution of the fitness through generations, with SOO, considering the first example on I. We observe a rapid growth of the fitness function in the first few generations, followed by a slower increase in the subsequent generations, which is a typical behavior of the evolution with GAs, indicating that convergence is being achieved. The average and the highest fitness grow roughly at the same pace — the blue and orange curves have, approximately, the same shape. Moreover, we see that our stop criteria seem reasonable, as it appears that the algorithm has already reached a plateau. Our intuition is that running more generations would produce no significant improvements (this hypothesis will be further discussed in the next section IV-D).

In order to best compare SOO and MOO final individuals, we projected them into 2-dimensional spaces, considering all pairs of two objective functions — **Figure 5** contains the best results for both optimization methods, considering the second configuration in Table I. The blue dots represent the final population generated with MOO, some of which have a colored circle around them, meaning that they are part of the first Pareto front. On the caption, the first number inside the identification [a, b] represents the generation in which each individual was generated, while b is an identifier. The red X represents the best individual generated with SOO for the same configuration.

Interestingly, SOO often performs as well as, or even better than, MOO. The red X frequently overlaps or surpasses the individuals on MOO's Pareto front. Formally, we can say that the SOO solution dominates the majority of MOO solutions. Specifically in **Figure 5**, the red X dominates all Pareto front solutions in some of the 2D projections. This tendency was not exclusive to this example but was observed in more experiments. The fact that SOO searches for solutions in a specific direction presents an advantage for our use case since we are seeking solutions that incorporate both the inspirations from the initial movies as well as user's instructions. MOO, on the other hand, will favor also extreme cases.

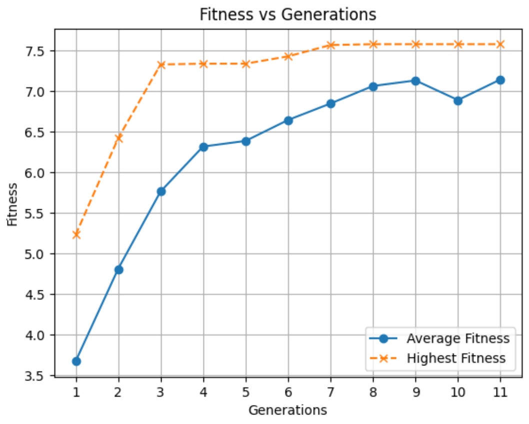

Fig. 6. Evolution of the fitness through the generations with $N_{gen} = 15$. After a few generations without improvement, our algorithm stopped before the $15^{th} gen$.

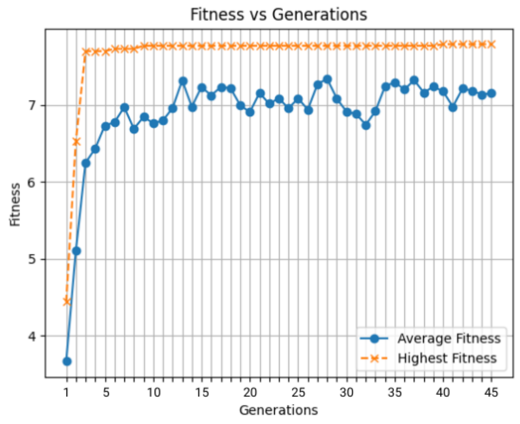

Fig. 7. Evolution of the fitness through generations with $N_{gen} = 45$, without any early stop criteria.

For example, a movie that scores higher than any other individual on the '*Instructions*' axis but very poorly on all other objectives will still be retained inside the final population. This occurred, for instance, with individual [1, 1] (red circle) in **Figure 5**, which persisted into the last population, despite not being a movie with all the requirements we are looking for. In fact, [1, 1] is a product of the initialization process where the LLM is asked to generate a movie according to the user's instructions, and that's why it fully satisfies the "Instructions" objective but not the remaining ones. The main reason why SOO seems more adequate is because we are looking for a good *equilibrium* in terms of the movie's content, and extreme cases do not serve that purpose. It is clear, however, that MOO may also produce well-balanced plots, figuring in a more central part of the Pareto front, but not at the extremes.

Finally, still regarding **Figure 5**, we can notice two more aspects. Firstly, '*Quality*' objective always gets scores around 7, with almost always between 6 and 8, an empirical confirmation of a phenomenon already discussed in Section IV-A. Secondly, when comparing the four objective functions, the '*Inspirations*' is the one that gets, on average, poorer scores – in **Figure 5** no individual scored more than 6 points. This is not exclusive to these examples but happened repeatedly. One possible explanation is that the '*Inspirations*' score is computed as an average of the inspirations from each initially selected movie, making it more difficult to achieve a high score, as it would require high inspiration scores across all the ancestors.

D. Increasing Complexity

In this section we tackle the following questions: is it worth running the algorithm with a larger number of generations? What if we increase the population size? What are the computational implications associated?

We considered the first configuration in **Table I** and ran it for 45 generations ($N_{gen} = 45$), without any stop criteria, both with SOO and MOO. Later, always with the same setup, we decided to run it with a population size of 45 ($N = 45$), but with the previous number of generations ($N_{gen} = 15$). We only changed one parameter at a time, so that we could analyze its effects independently of the other variables.

Table III summarizes this analysis. Note that, since the top fitness score is based on the weights used in SOO's equation, it inherently favors higher fitness scores for SOO than MOO.

Considering the scenario with $N_{gen} = 45$, we see no substantial improvement in the algorithm's performance. In fact, in **Figure 7**, which shows the evolution of the fitness function through the generations for the SOO case, we observe a clear plateau in the highest fitness over the generations, meaning that our algorithm is not benefiting from the increase in N_{gen}. When compared to the results with $N_{gen} = 15$, the increase in the highest fitness in the SOO case was only 0.21 which is negligible (see Table III). The MOO case was no better, once again, proving that the model had already arrived at a stable plateau only within a few generations, as seen in the previous scenario. Our intuition is that iterating over too many generations will drive away new individuals from the information contained in the 'inspirational' movies.

Secondly, the scenario with increased population size $N = 45$ is slightly more optimistic, although it does not show a drastic improvement. In SOO, the top fitness grew by 4.22% – higher than the previous value – but it is not significant when compared to the large increment in the population size and the consequent computational cost added. Still, we found that increasing N seems to affect positively the results even if the improvements are small.

Finally, we wanted to understand the potential benefits of using GPU parallel computing. With our computational power (CPU Intel Xeon Processor E5-2609 v4 with 64GB of RAM), we estimated that the generation and evaluation of 1 individual takes, in the worst case, ~ 24.34 seconds. This value refers to our current setup, without GPU parallelization, meaning that, if we parallelize the generation and evaluation of new individuals inside the same generation, we could use only

TABLE III
RESULTS – BEST SCORE AND TIME – WITH INCREASED N AND N_{gen}, COMPARED WITH THE PREVIOUS SETUP ($N = 15$ AND $N_{gen} = 15$). *MOO TOP FITNESS SCORE WAS COMPUTED BY TAKING A WEIGHTED SUM OF THE FITNESSES BY USING THE SAME VALUES CHOSEN FOR SOO.

	Top Fitness		Time (min)	
Setup	SOO	MOO*	SOO	MOO
$N_{gen} = 45$	7.79	7.65	193	189
$N = 45$	7.9	7.79	242	222
Previously	7.58	7.59	47	34

TABLE IV
AVERAGE VALUES FOR THE CORRELATION BETWEEN GPT AND HUMAN RANKINGS

	Human agreement w/		**Inter-human**
	GPT-3.5	**GPT-4o**	agreement
Inspirations	$\mathbf{0.04}$	$\mathbf{0.05}$	$\mathbf{0.42}$
Instructions	$\mathbf{0.53}$	$\mathbf{0.58}$	$\mathbf{0.33}$
Quality	-0.17	-0.14	$\mathbf{0.11}$

24.34 seconds/generation. This can lead to notable time savings, assuming that we have unlimited access to computational power. If applied to the scenario with $N_{gen} = 45$, the total time would become $45 \times 24.34 \approx 18\,min$. More, considering the case with fewer generations ($N_{gen} = 15$) but with larger N, the time savings would be even greater: $15 \times 24.34 \approx 6\,min$. These are substantial time savings, bringing more potential to our tool. In the scenario with 45 generations (which took 193 minutes), parallelization reduces the time to 18 minutes – a 90% time reduction!

E. Validating the Fitness Function

This section discusses the results obtained with the surveys described in Section III-D. In short, we asked people to rank a set of artificially generated movies considering the exact same criteria that the fitness function evaluates — this is, considering the inspiration in the chosen movies, the given instructions and the overall quality of the new plot. Ideally, if people agree with the LLM it would mean that the fitness function is well defined.

We obtained a total of 51 answers across six different surveys. Considering separately each survey, we measured both the agreement between different human evaluators and the human agreement with the GPT evaluation. In order to get an overall picture of the results obtained, we built **Table IV** that encapsulates the average results of the six surveys for each one of the objective functions. In particular, this table merges all 'inspiration' movies into one row, by computing their average value. Positive values, indicating a positive correlation, were highlighted in bold.

Notably, we observe that the third column (inter-human agreement) contains only positive values, reflecting a strong correlation between different human evaluators (Spearman's coefficient > 0). Note that this is important to validate our study, as lack of consensus among human evaluators would compromise the relevance of comparing human and GPT evaluations. Once we establish that there is generally a human consensus, we may proceed to evaluate weather GPT assessments correlate with the human rankings. Breaking down our analysis by the different objective functions, we observe the following:

1) Considering the 'inspiration' movies, the correlation factor does not show a clear pattern across the shown examples (Spearman coefficient ~ 0). Because this task is very dependent on the movies themselves, it is difficult to draw any general conclusion that fits most of the situations. In short, we found no correlation between human and GPT rankings for the '*Inspirations*' objective function;
2) Generally, human and GPT evaluations align when determining which movies best follow the given instructions (Spearman coefficient $\gg 0$). This is evident by looking at the positive values in the second row, indicating a positive correlation between both rankings in most cases. Our intuition is that the good agreement shown for this objective function has to do with the objectiveness of the task: either the given movie follows the given instructions or it does not, leaving less space for ambiguity;
3) When considering the '*Quality*' objective function, the correlation between human and GPT rankings is predominately negative, meaning that human and GPT ranking disagree often (Spearman coefficient < 0). One likely explanation is the limited validity of the GPT '*Quality*' evaluation, which often assigns scores close to 7 or 8 (see Section IV-A), making it difficult to distinguish high-quality movies from lower-quality ones. In addition, this row also shows the weakest correlation between GPT and human rankings — probably due to the subjectivity of the task — further contributing to the bad results observed.

Our findings indicate that there is a partial agreement between the GPT scores and the human feedback that we obtained. This suggests two possible interpretations:

1) The human feedback provided is not sufficiently accurate, so the discrepancies observed do not have validity nor meaning within the GA performance. This hypothesis is not as unlikely as it seems, given the nature of the surveys — they were exceptionally long, exhaustive, and featured stories that were very similar among them. This made the task quite complex, so we cannot guarantee that everyone has completed it with correctness.
2) Assuming that the human feedback has validity, the fact that the GPT's evaluation does not fully align with human rankings suggests that the fitness function in the GA may not be optimally guiding the evolutionary

process.

V. Conclusions and Future Work

We want to emphasize the disruptive nature of our work — we built this model from scratch combining knowledge from different Data Science paradigms in an yet barely unexplored domain, where creativity was key. Creativity — which is subjective, abstract and difficult to quantify — adds a layer of complexity to the model's design because it requires more than just technical accuracy — it demands generating content appealing to human sensibilities. The innovative nature of our approach presented us with many challenges, including dealing with unstructured data and evaluating highly subjective content. We are happy with the performance of our model — however, it is clear that a work that was developed from scratch in a short time, and of such nature will present limitations and there will be room for future improvement. The most important insights and limitations found in our work will be presented in this section.

A. Conclusions

We start by claiming that the architecture design described in Chapter III successfully worked, with movie plots evolving through generations, refining their traits and resulting in new movie ideas that meet the desired requirements. Our approach has shown great potential, especially if the algorithm is run with the operations in parallel (as described in Section IV-D), as it yields significant time savings.

In brief, we studied the influence of several parameters on the quality of the artificially generated movies. Here we list our main conclusions:

1) The choice of a robust LLM is crucial for the generation of quality movie plots, with GPT-4o outperforming GPT-3.5;
2) The initial setup, defining the list of existing movies that will serve as inspiration for our model together with user guidelines, also plays a decisive role in the quality of the generated content — however, those parameters are decided by users and, thus, are dependent on their creative philosophy;
3) When setting the optimization mode (SOO and MOO), we found out that the SOO solution often dominates the majority of MOO Pareto solutions, but not necessarily all of them;
4) Increasing the population size *slightly* improves the overall performance — we observed a 4.22% increase, when N was increased from 15 to 45;
5) A large increase in the number of generations produced a *negligible* improvement in the fitness of the best individual.

Our model still shows some limitations, namely when it comes to the quality of the generated movies. Often, some of the generated movies lack coherence, originality and describe very ambiguous scenarios or situations. Prompting engineering techniques may help to tune the LLM answers and to prevent those mistakes — however, those techniques only work to a certain extent, especially considering long and complex tasks like combining two movies. Moreover, logical inconsistencies are a frequent phenomenon, a type of LLM hallucination, and are difficult to solve. On a final consideration, we believe that our tool has enough potential to assist human writers in their creative process, as a way of blending stories and new ideas into a single new story. In summary, the limitations encountered were expected and inherent to any novel and exploratory approach like ours.

B. Future Work

As a part of future work, our methodology can be extended to a broader range of creative domains, such as novel writing, video game conceptualization, marketing copywriting, and other fields where creativity is key. To improve the user experience while using our tool, a logical next step could involve the development of an application with a user-friendly interface. In addition, processing operations in parallel, as described in Section IV-D, would drastically reduce the runtime, further improving user satisfaction. Moreover, future work could include simplifying the surveys, as section IV-E suggested that they may be too long and exhaustive. Finally, consulting a scriptwriting expert would provide valuable insights to help design a better and task-specific tool.

References

[1] Joel Lehman, Jonathan Gordon, Shawn Jain, Kamal Ndousse, Cathy Yeh, and Kenneth O. Stanley. Evolution through large models. 6 2022.

[2] Herbie Bradley, Andrew Dai, Hannah Teufel, Jenny Zhang, Koen Oostermeijer, Marco Bellagente, Jeff Clune, Kenneth Stanley, Grégory Schott, and Joel Lehman. Quality-diversity through ai feedback. 10 2023.

[3] Elliot Meyerson, Mark J. Nelson, Herbie Bradley, Adam Gaier, Arash Moradi, Amy K. Hoover, and Joel Lehman. Language model crossover: Variation through few-shot prompting, 2024.

[4] Pier Luca Lanzi and Daniele Loiacono. Chatgpt and other large language models as evolutionary engines for online interactive collaborative game design. In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '23. ACM, July 2023.

[5] Angelica Chen, David M. Dohan, and David R. So. Evoprompting: Language models for code-level neural architecture search, 2023.

[6] Can Xu, Qingfeng Sun, Kai Zheng, Xiubo Geng, Pu Zhao, Jiazhan Feng, Chongyang Tao, and Daxin Jiang. Wizardlm: Empowering large language models to follow complex instructions, 2023.

[7] Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. Self-refine: Iterative refinement with self-feedback. 3 2023.

[8] Herbie Bradley, Honglu Fan, Theodoros Galanos, Ryan Zhou, Daniel Scott, and Joel Lehman. *The OpenELM Library: Leveraging Progress in Language Models for Novel Evolutionary Algorithms*, pages 177–201. Springer Nature Singapore, Singapore, 2024.

[9] David Bamman, Brendan O'Connor, and Noah A. Smith. Learning latent personas of film characters. In Hinrich Schuetze, Pascale Fung, and Massimo Poesio, editors, *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 352–361, Sofia, Bulgaria, August 2013. Association for Computational Linguistics.

[10] Thomas W. MacFarland and Jan M. Yates. *Spearman's Rank-Difference Coefficient of Correlation*, pages 249–297. Springer International Publishing, Cham, 2016.