

Artificial Creativity: combining Evolutionary Computing with Large Language Models for creative scriptwriting

Rodrigo Silva Dias Lopes

Thesis to obtain the Master of Science Degree in

Data Science and Engineering

Supervisors: Prof. Nuno Cavaco Gomes Horta
Prof. Daniele Loiacono

Examination Committee

Chairperson: Prof. Mário Alexandre Teles de Figueiredo
Supervisor: Prof. Nuno Cavaco Gomes Horta
Member of the Committee: Prof. João Paulo Carvalho

November 2024

Declaration

I declare that this document is an original work of my own authorship and that it fulfills all the requirements of the Code of Conduct and Good Practices of the Universidade de Lisboa.

Acknowledgments

This thesis would not have been possible without the incredible support and encouragement I received along the way.

I would like to extend my gratitude to my supervisors, Professor Daniele Loiacono from Politecnico di Milano and Professor Nuno Horta from Instituto Superior Técnico for their guidance and mentorship throughout this project.

My deepest thanks go to all my family, especially my parents, for everything they have taught me throughout these years, for all their lessons and love, to my sister, for her patience and companionship, and to my grandparents, for their valuable friendship and unconditional support, all of you have shaped the person I am today.

To all my friends and loved ones, your positivity and good humor continually fuel my curiosity and passion for learning, thank you for your constant support and encouragement throughout this journey.

To everyone who helped shape this work in their unique way, all I can say is obrigado.

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.

Keywords

Large Language Models (LLMs); Evolutionary Computing; Genetic Algorithms; Evolution Through Large Models; Scriptwriting

Resumo

Este trabalho foca-se na integração de Large Language Models (LLMs) com Algoritmos Evolucionários (EAs), como uma ferramenta para escrita criativa, em particular no domínio da escrita de filmes. Este é um trabalho de natureza exploratória, com o objetivo de auxiliar guionistas (*users*) no seu processo criativo.

O trabalho envolve a geração de novos guiões para filmes, inspirados quer noutros filmes, quer em indicações dadas pelo *user*. Os guiões, neste contexto, podem ser conceptualizados como pedaços de texto que são gerados e manuseados pelo LLM, e que evoluem com o EA. O algoritmo genético (GA) irá recombinar, mutar, selecionar e avaliar os novos guiões, fazendo com que as ideias mais promissoras sejam privilegiadas. O LLM será inserido nas fases de recombinação, mutação e avaliação do GA. Dado que a fase avaliação utiliza diversas funções-objetivo, ambas as técnicas de *single-* e *multi-objective optimization* serão testadas, para os modelos GPT-3.5 e GPT-4o.

A nossa abordagem mostrou-se promissora, sendo que as ideias evoluíram e melhoraram ao longo das gerações, resultando em enredos inovadores que cumprem os requisitos mencionados. Testámos o nosso modelo dando sempre como input três filmes reais para inspiração, retirados de uma base de dados de 42.306 filmes. Verificámos que utilizar o GPT-4o com *single-objective optimization* e uma população grande produzia os melhores resultados. De modo a validar a avaliação dada pelo LLM, criámos um estudo onde recolhemos feedback humano. Em suma, apesar das limitações encontradas, este trabalho combina eficazmente dois paradigmas de Ciência de Dados, gerando conteúdo de natureza criativa.

Palavras Chave

Modelos de Linguagem de Grande Escala (LLMs); Computação Evolucionária; Algoritmos Genéticos; Escrita de Filmes

Contents

1	Introduction	1
1.1	Motivation	2
1.2	Goals	2
1.3	Achievements	3
1.4	Thesis Outline	4
2	Theoretical Background	7
2.1	Theoretical Background of Large Language Models	8
2.1.1	Machine Learning	8
2.1.2	Deep Learning	9
2.1.3	Large Language Models	11
2.1.3.A	Transformer Architecture	12
2.1.3.B	Attention Mechanism	15
2.1.3.C	Training a LLM	17
2.1.4	Prompt Engineering	17
2.1.5	Limitations of LLMs	19
2.2	Evolutionary Computing	20
2.2.1	Single and Multi-Objective Optimization	23
2.2.2	Genetic Algorithms	24
2.2.3	Elitist Non-Dominated Sorting Genetic Algorithm (NSGA-II)	25
2.3	Related Work	27
3	Methodology	31
3.1	Architecture Overview	32
3.2	Movie Plots Database	34
3.3	Initialization of the Evolutionary Algorithm	35
3.4	Recombination and Mutation Operators	37
3.5	Evaluation Operation	41
3.6	Selection Operation	45

3.7	Evolution with Single and Multi-Objective Optimization	45
3.8	Validating the Fitness Function: Human Feedback	46
3.9	Conclusions	49
4	Results and Discussion	51
4.1	Qualitative Analysis	52
4.2	Discussing Large Language Models and their Limitations	55
4.3	Analysis of the Evolution	56
4.4	Increasing Complexity	60
4.5	Fitness Function Assessment with Human Feedback	65
4.6	Conclusions	69
5	Conclusions and Future Work	71
5.1	Conclusions	72
5.2	Future Work	74
	Bibliography	75
A	Appendix A: Examples	81
A.1	Recombination	81
A.1.1	Recombination performed by GPT-3.5: <i>Mamma Mia!</i> + <i>Back to the Future</i>	81
A.1.2	Recombination performed by GPT-3.5: <i>Mamma Mia!</i> + <i>Toy Story</i>	82
A.1.3	Recombination performed by GPT-4o: <i>Mamma Mia!</i> + <i>Back to the Future</i>	83
A.2	Mutation	84
A.2.1	Mutation performed by GPT-3.5: <i>Mamma Mia!</i>	84
A.2.2	Mutation performed by GPT-4o: <i>Mamma Mia!</i>	85
A.3	Evaluation	86
A.3.1	Evaluating Quality f_Q with GPT-3.5	86
A.3.2	Evaluating Quality f_Q with GPT-4o	87
A.3.3	Evaluating Inspirations f_{insp} with GPT-3.5	88
A.3.4	Evaluating Inspirations f_{insp} with GPT-4o	88
A.3.5	Evaluating Instructions f_{inst} with GPT-3.5	89
A.3.6	Evaluating Instructions f_{inst} with GPT-4o	89

List of Figures

2.1	Perceptron	10
2.2	Architecture of a feed-forward Neural Network (NN)	10
2.3	Overview of the transformer decoder-only architecture for text generation.	14
2.4	Pareto Fronts highlighted with a darker line. The search space is identified with a lighter color.	24
2.5	NSGA-II architecture	26
3.1	Architecture overview of the developed work.	32
3.2	Example of an individual: a free-form piece of text containing the movie's description and its summary.	33
3.3	Example of a user interaction with our application.	35
3.4	Initialization Prompt. Signed in blue, the user's instructions should be inserted.	36
3.5	Initialization of the population, inspired by the chosen movies and user's prompt.	37
3.6	Structure of the recombination/mutation prompt, including role assignment and few-shot learning.	38
3.7	User Message in the context of Recombination. Signaled in blue, the actual movies to be combined are inserted.	39
3.8	User Message in the context of Mutation. Signaled in blue, the actual movie to be mutated is inserted.	40
3.9	User Message inside the prompt to evaluate the plot's quality.	42
3.10	User Message inside the prompt to evaluate how much the new plot is inspired by the chosen movies.	43
3.11	User Message inside the prompt to evaluate how much the new plot follows the user's brief.	44
4.1	Setup No.3: Comparing Single-Objective Optimization (SOO) and Multi-Objective Optimization (MOO) solutions in the objective space.	57
4.2	Setup No.4: Comparing SOO and MOO solutions in the objective space.	58

4.3	Evolution of the fitness through generations with Single-Objective Optimization.	60
4.4	Comparing SOO and MOO solutions in the objective space with $N_{gen} = 45$	61
4.5	Comparing SOO and MOO solutions in the objective space with pop. size $N = 45$	62
4.6	Evolution of the fitness through generations with Single -Objective Optimization with N_{gen} and $N = 45$	63
4.7	Computing rank correlation between Human and GPT evaluations	66
4.8	Computing Inter-human Agreement	66

List of Tables

2.1	Related Work overview	29
3.1	Setting up the generation of new movies for the surveys. 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.	50
4.1	Table with the setting of parameters for the experiments done to analyze evolution.	56
4.2	Results – best score, time and last generation number – considering six different initial setups.	56
4.3	Results – best score and time – for an $N = 45$ (population size) and $N_{gen} = 45$	61
4.4	Average time for each operation	64
4.5	Number of answers per survey	65
4.6	Survey No.1	67
4.7	Survey No.2	67
4.8	Survey No.3	67
4.9	Survey No.4	67
4.10	Survey No.5	67
4.11	Survey No.6	67
4.12	Average values for the correlation between GPT and human rankings	68

Acronyms

AI	Artificial Intelligence
CD	Crowding Distance
CNN	Convolutional Neural Network
CoT	Chain of Thought
DL	Deep Learning
EA	Evolutionary Algorithm
ELM	Evolution through Large Models
GA	Genetic Algorithm
GAN	Generative Adversarial Network
LLM	Large Language Model
LMX	Language Model Crossover
ML	Machine Learning
MOO	Multi-Objective Optimization
NLP	Natural Language Processing
NN	Neural Network
NSGA-II	Elitist Non-Dominated Sorting Genetic Algorithm
QDAIF	Quality-Diversity through AI Feedback
RNN	Recurrent Neural Network
SOO	Single-Objective Optimization

1

Introduction

Contents

1.1	Motivation	2
1.2	Goals	2
1.3	Achievements	3
1.4	Thesis Outline	4

The main motivations for this work will be presented in Section 1.1, its purposes and goals in Section 1.2, and finally a summary of its achievements in Section 1.3. One should get a clear perspective on what is the objective of this work and an overall overview by reading this Chapter.

1.1 Motivation

Creativity was always seen as an inherently human capability. It is the source of every human creation: each book, each game, each song, or even each joke would not have been possible without this tool that we call *creativity*. Nevertheless, creativity is not always easy to achieve and many times people struggle at finding it. Fortunately today, thanks to the rise of Artificial Intelligence (AI), creative tasks can be performed with the help of technology.

The scope of this work is to construct a tool that helps people perform creative tasks. In fact, the key idea that motivates our work is the potential to evolve ideas through an Evolutionary Algorithm (EA). In particular, this work will focus on one particular task: **writing movie plots**. 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) into this work. Also, the fact that these objects 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.

Note that this work should be seen as a helpful tool for writers and creators but not as a replacement for the human role. Creativity is still inherently *human* (as stated before), in the sense that the tools that use Artificial Intelligence (AI) do nothing but mimic human creativity.

1.2 Goals

This work aims at combining Evolutionary Computation with a Large Language Model (LLM) in order to generate new ideas, in particular of movies, based on previously existing ideas. The Evolutionary Algorithm (EA) will allow us to explore different ideas and determine which ones should be further exploited, while the LLM is crucial at handling and manipulating the movie descriptions, as they are simply a piece of text, and thus, easily processed by the LLM.

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. The evaluation process is pivotal in guiding the Evolutionary Algorithm (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 agreement 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 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.

1.3 Achievements

In this work, we successfully built an assisting tool for generating movie plots. The tool requires the following inputs from the user:

- **Titles of real movies:** Users can provide an arbitrary number of movie titles that will serve as inspiration for generating the new plots (e.g. "The Matrix", "Back to the Future", ...);
- **(Optional) Custom guidelines or instructions:** Users may also provide specific directions to guide the plot generation (e.g., "Write a dystopian movie set in the near future.").

The tool will, firstly, retrieve the summaries of the provided movies from a database of 42,306 films [1]. These summaries, together with the initial user instructions, will evolve through the Genetic Algorithm (GA), where variation and evaluation operations are performed recurring to a LLM. Besides the aforementioned inputs, the user can also customize the output by adjusting some default parameters, including the selected language model and parameters specific to the GA. The **final output**, corresponding to the GA population at the last generation, is a set of original, detailed and structured movie plots. Each new movie plot includes:

- **Title**
- **Genre**
- **World-building description**
- **List of the main characters**
- **Plot divided into three acts**, following the traditional three-act movie structure.

This tool outputs a set of original movie ideas that align with the initial inputs, offering flexibility and innovation in the scriptwriting creative process.

While LLMs alone (like ChatGPT) can be an efficient tool during the creation process, we emphasize that **the primary goal of this project** was to experiment with the possibility of **evolving ideas** through a Genetic Algorithm (GA). The outcome of our study was **positive**, as we observed movie ideas evolving and improving through the generations, with the GA refining the best ideas while discarding the worst ones.

1.4 Thesis Outline

The next chapter, Chapter 2, delves into the theoretical background supporting our work. We begin by explaining the foundational theory behind **Large Language Models (LLMs)**, starting from the concepts of Machine Learning (ML) and culminating in the Transformer architecture, the foundation of modern LLMs. This is followed by a discussion on the usage and limitations of LLMs, along with a section exploring prompting techniques. The second part of Chapter 2 dives into the theory of **Evolutionary Computing**, covering the topics of single- and multi-objective optimization, defining Genetic Algorithms (GAs) and featuring an explanation on the Elitist Non-Dominated Sorting Genetic Algorithm (NSGA-II). Finally, 2.3 explores other **related works** that are, somehow, connected to this thesis by, at least, one of its three main pillars: evolutionary computing through Large Language Models (LLMs); the use of LLMs to provide feedback on the generated content; and the usage of AI as a tool for creative writing.

Chapter 3 is focused on explaining in detail the **architecture and methodology** of our work. It begins by providing an overview of the global project architecture on 3.1, followed by an in-depth explanation of each one of the sub-parts or modules that constitute the overall architecture, from Sections 3.2 to 3.8. This Chapter includes not only descriptions of every step and every operation of the GA, but also a description of the study we conducted to collect human feedback in an attempt to assess the validity of the fitness function.

Chapter 4 presents, analyses and discusses the results obtained in this work. It aims at providing a **concrete, unbiased analysis of our findings**, highlighting both its successes and limitations. We start with a qualitative analysis of the artificially generated movie plots (4.1) and a comparative analysis of the results obtained with different LLMs (4.2). Sections 4.3 – 4.4 focus on the proprieties of the evolution with the GA, specially, its optimization modes and the potential benefits of increasing complexity to understand the scalability of our model. Finally, Section 4.5 provides a clear analysis of the results obtained with the collected human feedback. The analyses and findings in this chapter are respectively supported by numerical data, graphs, plots, tables and suitable examples.

Chapter 5 concludes by summarizing all the **key findings** of this project and explores potential

directions for **future development** of our work.

Additionally, Appendix A contains some **examples of text generated** by the LLM during the evolutionary process, supporting the qualitative analysis of Chapter 4.

2

Theoretical Background

Contents

2.1 Theoretical Background of Large Language Models	8
2.2 Evolutionary Computing	20
2.3 Related Work	27

This chapter presents the theoretical foundations that support our work. We begin by exploring the principles of ML and Deep Learning (DL) that gave birth to the Large Language Models (LLMs) (Section 2.1). Next, we present Evolutionary Computing theory, focusing on the concepts behind Genetic Algorithms (GAs) (Section 2.2). Finally, we analyze the State-of-the-Art (Section 2.3), highlighting recent developments in the fields most relevant to our work.

2.1 Theoretical Background of Large Language Models

In order to understand the origin of LLMs, it is necessary to dive first into the basic concepts that lie behind it. Namely, we shall discuss in this Section the field of **Machine Learning** 2.1.1, which has later given rise to **Deep Learning** 2.1.2 - due to the development of **Neural Networks**. Later in this Section, we will explore in more detail what are the mechanisms that led to the rise of **Large Language Models** 2.1.3 (namely, the Attention mechanism) and finally discuss some **Prompt Engineering** 2.1.4 techniques used today.

2.1.1 Machine Learning

According to the definition given in 1959 by Arthur Samuel, one of the pioneers in this field, ML is "the field of study that gives computers the ability to learn without being explicitly programmed." [2] In fact, Machine Learning refers to a group of algorithms and statistical models that are able to *learn* from experience by identifying patterns underlying specific types of data. Once the algorithm has *learned* the pattern, it is capable of generalizing, allowing for accurate predictions about new, unseen data. The *learning* process involves optimizing a performance metric, meaning that the ML algorithm should improve its performance on a specific task over time [3].

Machine Learning can be subdivided into three main categories:

1. **Supervised Learning:** learning from examples. The learning process involves finding a mapping function between the input data and a desired (known) output. The model learns iteratively, meaning that it will adjust its parameters in order to minimize the difference between the predicted and the desired outputs on a training dataset. After training, the model should be capable of generalizing to unseen data, accurately predicting the output of new inputs. To sum up, a supervised learning model should mimic the underlying pattern that correlates the input and output data, by observing numerous labeled examples. [3]
2. **Unsupervised Learning:** aims at discovering patterns and structures - such as probability distributions or clusters - within an unlabeled dataset, without any specific guidance. The main goal is to

extract meaningful information and insights from data in an exploratory manner, without any supervision. Examples of unsupervised learning algorithms include clustering, dimensionality reduction, and others. [3]

3. **Reinforcement Learning:** is a technique where a computer agent learns how to perform a certain task based on feedback from the environment. The agent makes decisions with the objective of maximizing a reward metric, receiving feedback from the environment in the form of a penalty or a reward. This means that the agent learns with experience, in a trial-and-error process, without the need of being explicitly programmed. [4]

In this work, we will focus particularly on the first type of ML, since Supervised Learning is the dominant paradigm for the development of LLMs. In fact, the training process of an LLM involves trying to predict correctly labeled data. For instance, if the goal is to predict the next word in a sentence, the predicted word should ideally match the actual word in the dataset, otherwise the performance metric should be penalized.

2.1.2 Deep Learning

DL is a sub-field of Machine Learning that use deep Neural Networks (NNs) to make predictions or take decisions based on knowledge acquired from large amounts of data. The adjective *deep* refers to the multiple layers used in the architecture of these NN. Let's dive into a more detailed explanation.

The history of the development of NN goes back to the invention of the perceptron algorithm in 1943 by Warren McCulloch and Walter Pitts. Interestingly, with McCulloch being a neurophysiologist and Pitts being a logician, the perceptron (also known as McCulloch-Pitts neuron or artificial neuron) attempts to mimic the behavior of the biological neuron in a very simplified way (with emphasis on its simplified representation) [5].

The **perceptron** - which belongs to the class of supervised learning algorithms - is a linear classifier that can be used to solve linearly separable problems (Figure 2.1). It works by computing a linear combination between the input values and the weights associated with each one of them. The result is then subject to an activation function which was originally set to be the Heaviside step function - meaning that the neuron would 'fire' whenever the result of the linear combination exceeds a certain threshold. Later models replaced the hard threshold with more general activation functions, which include linear activation, sigmoid, hyperbolic tangent, softmax, and others. The weights are adjusted whenever the output of the perceptron and the label for that specific input do not coincide. [6]

The big limitation of the original perceptron algorithm is that it cannot solve non-linearly separable problems (such as the XOR problem, for example), unless these problems are transformed to a different representation that makes them linearly separable [7]. The way to overcome this limitation is to construct

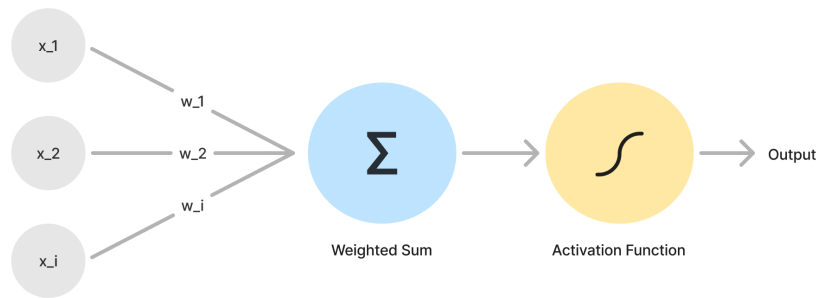


Figure 2.1: Perceptron

a **multi-layered perceptron** (also referred to as a **feed-forward neural network**), where many artificial neurons are stacked in each layer and will serve as the input vector to the next layer. Actually, a feed-forward neural network with just one hidden layer and a linear output can approximate arbitrarily well any continuous function, given enough hidden units (artificial neurons). This is known as the **Universal Approximation Theorem** and was originally proved by Hornik et al. in 1989 [8]. However, there is a caveat with having just a single hidden layer which is that one may need exponentially many hidden units and the problem may become computationally impracticable. On the other hand, deeper networks (NNs with many hidden layers) provide much more compact approximations [9].

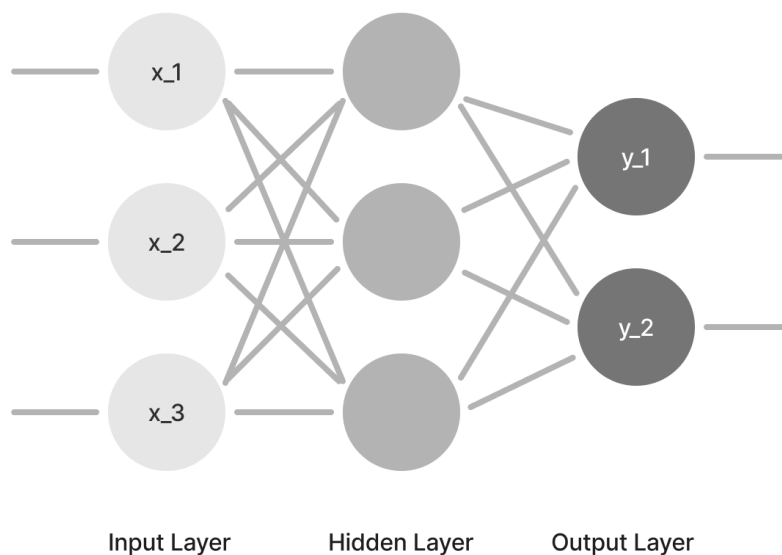


Figure 2.2: Architecture of a feed-forward NN

A feed-forward network is fully-connected (Figure 2.2), meaning that each neuron is connected to every other neuron in the adjacent layers and the strength of each connection is determined by the

corresponding weight. Typically, the network is trained with a very large number of labeled examples (pairs of input-output vectors corresponding to the desired output of the network). During the training phase, an input vector is fed to the network and the output is computed. The prediction error is used later to adjust the weights by using an algorithm called **backpropagation**. Training involves applying backpropagation many times over several epochs, each epoch containing many batches (subsets of the training set). The goal of backpropagation is to minimize the loss function through gradient descent-based optimization algorithms. This allows the network to uncover relevant patterns in the training data and make correct predictions for the test data. Usually, a validation set of data is used before testing for hyper-parameter tuning (setting parameters other than the weights - for example, the number of hidden layers, the optimizer, etc).

Above we described how a perceptron by itself cannot solve non-linearly separable problems unless the input vector suffers a transformation that maps it to a new space where it becomes linearly separable. In general, this step of mapping input features to new (more useful) representations is called feature extraction. In the traditional Machine Learning (ML) flow, the input vectors are subject to a manually engineered process of feature extraction (also referred to as handcrafted features), which then serves as input to the ML algorithm. These handcrafted features are usually a way of encoding prior knowledge about the specific problem. Contrarily, by using a Deep Learning (DL) network, these 'useful' features are *learned* in the intermediate layers, without the need to be manually engineered. Typically, the initial layers will contain basic, low-level representations of the input vector, while the last layers should be able to have a more sophisticated, high-level representation of the input [10].

Feed-forward Neural Networks are one of the several NN architectures that have been developed. Some of the most common architectures include Recurrent Neural Networks (RNNs), Convolutional Neural Networks (CNNs), Generative Adversarial Networks (GANs), Autoencoders and Transformers, but there are many other architectures. LLMs are based on the transformer architecture and we will explore it in more detail in the next section 2.1.3.

2.1.3 Large Language Models

Recent advancements both in Deep Learning (DL) and Natural Language Processing (NLP) allowed for the development of Large Language Models (LLMs), which, in simple terms, are deep learning models trained in enormous corpus of text that can understand natural language and generate human-like text responses. They are based on the transformer architecture, initially proposed in 2017 by a Google research team in a paper called "Attention Is All You Need". Indeed, the *attention* is the key mechanism that distinguishes transformers from other architectures [11].

Before diving into the details of transformers and the attention mechanism, it is important to mention the vast applications that LLMs can be used for inside the NLP universe. LLMs are able to perform a va-

riety of NLP tasks such as named entity recognition, sentiment analysis, translation, text summarization, question-answering, text completion, and more. Their ability to generate coherent and context-aware text makes them a powerful tool, mimicking humans in conversational and writing tasks.

Furthermore, the ability to generate new text means that LLMs fall into the category of Generative AI - which includes essentially all AI models that can create something new (either text, audio, images or video).

2.1.3.A Transformer Architecture

There are several families of LLM architectures based on transformers. Here, I will focus uniquely on a specific architecture called *decoder-only*, since all the LLMs used in this work implement a decoder-only based architecture. Namely, the two LLMs used in this thesis (**GPT-3.5** and **GPT-4o**) are based on the GPT decoder-only architecture. Because of that, I will use the largest version of GPT-3 model to illustrate all the examples in this section. The explanation given in this section is based on [12].

The task of **text generation** - the one we are interested in - can be conceptualized as the task of predicting the next word in a sentence and doing it iteratively, always updating the sentence by adding the word predicted in the previous step. For this reason, here we shall focus on how transformers perform the task of predicting the next word.

To begin with, the input sentence is chunked into tokens in a process known as **tokenization**. A token is the smallest semantic unit and it can be a word or a part of it. The exact tokenization rules vary with the chosen model. As an example, see the following phrase split into tokens (according to GPT-3 rules) [13]:

```
Token|ization| is| the| process| of| split|ing| a| sentence| into| smaller| parts|.
```

Afterwards, each one of these tokens are transformed into a high-dimensional vector known as an **word embedding** (the most popular term in literature is *word* embedding, but keep in mind that these embeddings actually refer to tokens). These embeddings come from the embedding matrix W_E , which is a matrix that contains the embeddings of all possible tokens. The size of W_E is, thus, embedding size \times size of the vocabulary (all possible tokens). Again, in the case of GPT-3 this means $12,288 \times 50,257$, since its vocabulary contains $\sim 50k$ tokens and each token is represented by a 12,288-dimensional vector [14]. Having tokens transformed into vectors aims at encoding their meaning into the embedding vectors. This means that, considering that vectors encode meaning, words (or tokens) with similar meanings stay close to each other in the multi-dimensional embedding space. Furthermore, this multi-dimensional embedding space should ideally be able to capture relationships between words. The following is the classic example in NLP literature [15]:

$$Man - Woman \approx King - Queen$$

In the above example, the multi-dimensional vector that comes from computing the difference between the embeddings of *man* and *woman* is very similar to the vector obtained if the difference is computed between *king* and *queen*. One can think of it as if both vectors are pointing in the same direction in the multi-dimensional embedding space, capturing the concept of gender moving from masculine to feminine. This example is just an illustration on how embeddings may encode relationships between words and may not represent the real relations that happen inside the model's embedding vectors. As a note, it is important to mention that word embeddings inside transformers also encode the position of the word in the text input. The technique used to represent the position of a token is known as positional encoding [16].

The embeddings of the tokens of the input text are then fed to the network. Each LLM has a fixed limit for the number of tokens that can be sent to the network in order to make the prediction, known as the **context window size**. For GPT-3, the context size is equal to 2,048 [17].

In a transformer decoder-only architecture, these embeddings follow a sequence of layers, alternating between **attention blocks** (explained in detail in subsection 2.1.3.B) and **feed-forward neural networks**. The hope is that after all these operations within the attention and the feed-forward blocks, the meaning inside each word embedding is refined, having received information about its surrounding context. As an illustration, consider the sentence "I can't find a spot to park my old car." By allowing the model to have access to the context, the hope is that the LLM understands better the meaning of each word and even more abstract concepts, such as the fact that "park" refers to the verb, and not to a green area. More abstract concepts include updating the vector representing "car", knowing the fact that it is *mine* and it is *old*.

The further down we go in this series of attention and feed-forward blocks, the hope is that we encounter more nuanced and higher-level concepts encoded into each one of the embeddings. The output of this sequence of blocks is a set of vectors representing the *new* embeddings with each one associated with each input token.

To recap, the initial token embeddings pass through a series of alternating attention and feed-forward blocks, always refining its encoded meaning. As we are focused on the task of predicting the next word (next token, more precisely), the final step consists of finding a **probability distribution** over all the possible tokens that might come next. This is done by multiplying another matrix known as the unembedding matrix W_U by the last embedding (the one corresponding to the last token). The size of W_U corresponds to the size of the vocabulary \times embedding size and by multiplying it by the last embedding we get a new vector with the size of the vocabulary. We call this new vector the **logits** vector. Indeed, each entry (also called a logit) is associated with one possible token, in such a way that higher values imply a higher probability of that token being the next. In order to map this vector to a

probability distribution, the **softmax function** is applied to it. Then, the next word is sampled from the obtained probability distribution.

Below, the softmax function is applied to a column vector:

$$\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{bmatrix} \xrightarrow{\text{Softmax}} \begin{bmatrix} e^{x_1/T} / \sum_{j=1}^N e^{x_j} \\ e^{x_2/T} / \sum_{j=1}^N e^{x_j} \\ \vdots \\ e^{x_N/T} / \sum_{j=1}^N e^{x_j} \end{bmatrix}$$

The output of the softmax can be interpreted as a probability distribution, as all entries are between 0 and 1 and they sum up to 1. The parameter T , known as temperature, is related to the strength of each logit in the final distribution. A higher T will produce a more uniform distribution, making the next word prediction more random. On the other hand, if T is set to zero, then the probability distribution will assign 1 to the highest logit, meaning that the predicted word will always be the one with the largest logit.

Figure 2.3 is a simplified illustration of the main components of the architecture of a decoder-only transformer. The task is to predict the next token given the initial sentence "I can't find". The sentence is tokenized into "I | can | 't | find" and transformed into the embeddings \vec{E}_1 to \vec{E}_4 that are then fed to the network. \vec{E}'_1 to \vec{E}'_4 are the transformed embeddings produced by the network. The predicted word would be "my", since it is the one with the highest probability after the softmax. The pair of attention and feed-forward layers repeat N times.

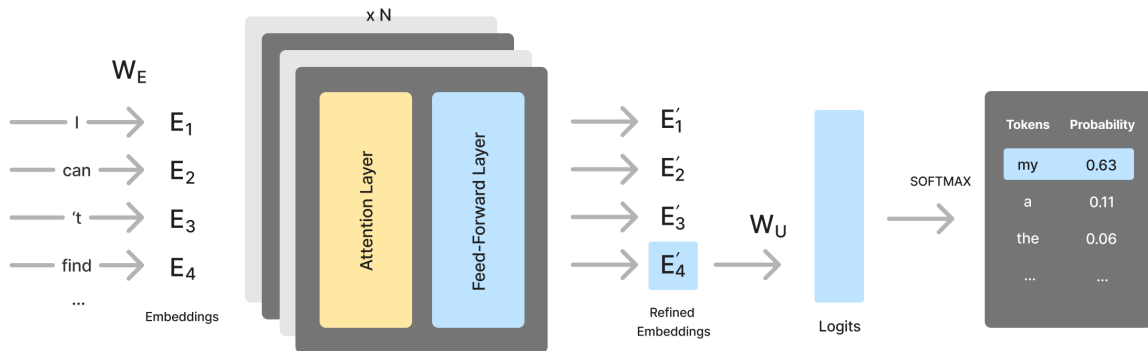


Figure 2.3: Overview of the transformer decoder-only architecture for text generation.

2.1.3.B Attention Mechanism

The attention blocks are the main factor responsible for the improvements in quality of the embeddings, especially because they focus on the context words to update the meaning of each embedding. The attention block is made of multiple attention heads. Here, we will do an overview on how attention works inside a transformer, based on [16, 18].

Keys, Queries and Values

Each embedding \vec{E}_i is associated with three vectors known as the key \vec{K}_i , query \vec{Q}_i and value \vec{V}_i vectors. These are obtained after multiplying three new matrices (respectively, W^K , W^Q and W^V) by the row vector of the embedding \vec{E}_i . These matrices will map the high-dimensional embedding vector to a lower-dimensional space. In the case of GPT-3, where the embedding size is 12,288, the new \vec{K}_i , \vec{Q}_i and \vec{V}_i vectors will be 128-dimensional [14]. Note that W^K , W^Q and W^V (respectively known as the key, query and value *weight* matrices) **are learned** during the training process.

$$\begin{aligned}\vec{E}_i \begin{bmatrix} & W^K \end{bmatrix} &= \vec{K}_i \\ \vec{E}_i \begin{bmatrix} & W^Q \end{bmatrix} &= \vec{Q}_i \\ \vec{E}_i \begin{bmatrix} & W^V \end{bmatrix} &= \vec{V}_i\end{aligned}$$

Attention Pattern

The attention pattern is calculated by multiplying the queries matrix Q by keys K matrix, by computing QK^T , where: Q contains all the query vectors in its rows; K^T contains all the key vectors in its columns. All entries of QK^T are then divided by \sqrt{d} , where d is the dimension of the key/query spaces, and then the **softmax function** is applied, because these attention scores will act as weights afterward.

These attention scores tell us essentially how relevant is each word to every other word inside the context window. More precisely, the attention scores evaluate how relevant is each *key* to update the meaning of each *query*. However, because the model is trying to predict the *next* word in a sentence, it does not make sense to evaluate the impact of a *future* word, which means that a mask will be applied to the upper right triangle of the QK^T matrix, in a process called masking.

$$QK^T = \begin{bmatrix} - & Q_1 & - \\ - & Q_2 & - \\ & \vdots & \\ - & Q_d & - \end{bmatrix} \begin{bmatrix} | & | & & | \\ K_1^T & K_2^T & \dots & K_d^T \\ | & | & & | \end{bmatrix} \xrightarrow{\text{softmax} + \text{masking}} \begin{bmatrix} \bullet & 0 & 0 & \dots & 0 \\ \bullet & \bullet & 0 & \dots & 0 \\ \bullet & \bullet & \bullet & \dots & 0 \\ \vdots & & & & \\ \bullet & \bullet & \bullet & \dots & \bullet \end{bmatrix}$$

Above, the attention pattern is represented by the black dots, where larger dots indicate larger attention scores. Each dot represents a value from 0 to 1 because softmax was applied, and the upper right triangle was set to zero due to masking. This is intended to be an illustration of the attention map – the size of the dots was imagined. A high attention score means that somehow the key is very relevant to update the meaning of that specific query, while a low score means that the two words associated with the key and query are unrelated. This step is crucial because it is the only step where the embeddings 'look' to their neighbors. The set of attention scores is called attention pattern.

Self-Attention

Once the attention pattern is computed, the attention scores will serve as weights to (finally!) create newly updated embeddings that encode – hopefully – a more refined meaning given the information collected from the neighbor tokens. It works by multiplying the attention scores by the value vector:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d}}\right)V = Z \quad (2.1)$$

V contains all the value vectors in its rows. The output of self-attention Z is a new matrix with as many rows as the number of embeddings in the context window and as many columns as the columns in the V matrix.

On a clarifying note, the terminology *self-attention* refers to the attention mechanism computed with respect to the same sequence, which is the case in our 'predicting the next token' example. Other tasks, like language translation, need to compute attention with respect to two different sequences and, hence, are not referred to as self-attention.

Multi-Head Attention

Each attention layer is made of many attention heads, hence the name multi-head attention. The advantage of using multi-head attention is that it gives the attention layer the chance of representing tokens in multiple 'subspaces', which helps to figure out the several complex relations that can exist between tokens in a sentence. As an example, GPT-3 uses 96 attention heads inside each attention block [14].

If, inside each attention head, we use different weight matrices, we will end up with many Z_i matrices that show up after computing self-attention in each head. All these matrices are concatenated into a

single big matrix, with as many rows as the number of embeddings and as many columns as the number of attention heads times the number of columns in the V matrices. This big matrix is then multiplied by another matrix W_O that maps the output of all attention heads to a space with the embedding dimension. W_O is known as the output matrix and its weights are learned during training.

$$Z' = \begin{bmatrix} Z_1 & | & Z_2 & | & \cdots & | & Z_N \end{bmatrix} \begin{bmatrix} W_O \end{bmatrix}$$

The result Z' has a size number of embeddings in the context window \times embedding dimension. Again, for GPT-3, this is $2,048 \times 12,288$. Finally, once multi-head attention is computed, the result will be fed to the feed-forward network.

2.1.3.C Training a LLM

The process of training an LLM involves adjusting billions of parameters, using very large amounts of training examples. The largest version of GPT-3 has 178 billion parameters [14]. Both the dimension of the training data and the dimension of the model (i.e., the number of parameters) have a large impact on the performance of the model, with bigger models being trained with more text data performing generally better [17, 19]. More precisely, these adjustable parameters are nothing other than the weight matrices. These include:

- W_E and W_U (embedding and unembedding matrices);
- W_Q , W_K and W_V , different for each attention head;
- W_O (output matrix);
- All the weights from the feed-forward layers.

It's common to say that these weights are *learned* during the training phase.

2.1.4 Prompt Engineering

There are several prompting techniques designed to guide the LLM to produce a certain desired output. These techniques involve crafting the *prompt* (input text) that is given to the LLM in a way that encourages the model to reason better and consequently yield a more accurate answer. Some basic rules for good prompting include providing detailed instructions using clear language, specifying the desired

output format, and adding clear syntax. Below, we will explore some of the most well-known prompting techniques [20–23].

Role Assignment involves structuring the prompt into three different parts, known as roles, which are the `System`, `User` and `Assistant`. Models like the **GPT** family (used in this thesis) offer the possibility of dividing the prompt message into these three different roles.

The `System` role (which is optional) comes right at the beginning and it was conceived to give the possibility to instruct the model to *act* in a certain way, guiding its response in terms of tone, style and depth of the explanations. Examples of the `System` message can vary from the classical examples to more creative ones, such as:

```
System: You are an helpful AI assistant.
```

or

```
System: Act like a food critic, who always writes using rhymes.
```

The `User` message corresponds to the actual instructions given by the user; while the `Assistant` message refers to the respective answer given by the model. Note that the `User` message is always required; on the other hand, the `Assistant` role will be used only in specific techniques, as it is the case of few-shot prompting (see below).

Zero-shot prompting is the most basic approach to prompt a model. It consists simply in assigning a task to the model and asking for results, without further providing any additional examples or context. Regarding the following example, we expect that the model responds `Positive`, ideally.

```
User: Classify the text into negative or positive: "I love ice cream."
```

```
Assistant:
```

Few-shot prompting consists, in contrast to zero-shot prompting, in providing the model with a few examples that illustrate the desired behavior of the model. The following example illustrates how this technique enables the model to better understand human intention, just by providing a couple of examples. Even without the explicit instruction (`Classify the text into negative or positive`), the model should be able to understand the underlying task and ideally answer `Negative`. However, it comes with more computational cost, as it spends more tokens for each prompt made.

```
User: "I love ice cream."
```

```
Assistant: Positive.
```

```
User: "The movie was quite boring."
```

Assistant: Negative.

User: "The show disappointed me."

Assistant:

Chain of Thought (CoT) is used for more complex tasks. It aims at breaking down the task into smaller tasks, encouraging the model to reason step-by-step, instead of prompting it with the entire task. In other words, it instructs the model to perform one step at a time, eventually leading to the final answer. The following example consists of **few-shot Chain of Thought**, since it provides an illustrative answer to exemplify how to reason step-by-step [24]:

User: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can have 3 tennis balls. How many tennis balls does he have now?

Assistant: Roger started with 5 balls. 2 cans of 3 tennis balls each is 6 tennis balls. $5 + 6 = 11$. The answer is 11.

User: The cafeteria had 23 apples. If they used 20 to make lunch and bought 6 more, how many apples do they have?

Assistant:

On the other hand, one can prompt the model with CoT without providing examples, just by asking the model to reason step-by-step - known as **zero-shot Chain of Thought**. In fact, Kojima et al. (2022) [25] showed that simply adding sentences that stimulate reasoning chains, like "Let's think step by step", to the end of the prompt actually increases the performance of the model in complex tasks. As an example:

User: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?

Assistant: Let's think step-by-step. (The model is supposed to continue from here, breaking down the problem into steps).

To conclude, these are some of the most commonly used prompt engineering techniques and the ones that are implemented in the context of this work. Other techniques include Self-Consistency, Generated Knowledge, ReAct, Retrieval Augmented Generation, among many others [21].

2.1.5 Limitations of LLMs

Large Language Models are a powerful tool able to generate text, acting almost like a human being. Nevertheless, one should remember that these tools do nothing other than predicting the next word. This

means that, even though they can understand language and its mechanisms like grammar and syntax, LLMs are not guaranteed to give factually correct or logical answers. They do not possess a real-world understanding about the semantics of words nor the relations between them. All the meanings acquired by the LLMs were extracted during the training process, meaning that they are simply based on huge corpora of text. There is no connection between the words seen by the model and the actual tangible objects or real-world events that they refer to, making it impossible for the model to have true knowledge of meaning [26].

This being said, there are several limitations of LLMs. Burstev et al. [27] argued that there are four important limitations, that they categorized as **limitations** of:

1. reasoning: refers to the incapability to solve logical problems due to the lack of understanding of logical rules.
2. knowledge or expertise: is associated with the undesired possibility of providing factually incorrect information or fabricating new information. Moreover, this limitation can lead to inconsistencies in the produced text, with the model answering contradictory things with respect to the same prompt.
3. understanding: happens when the the model seems to not understand the question itself or when it focuses on the wrong part of the problem.
4. planning and execution: refers to the incapability of LLMs to actually take real-world decisions and plan practical solutions, because they can not be fully trusted.

Moreover, the LLMs also suffer from **hallucinations**. Some of the limitations above mentioned consist of hallucinations. These can be classified into intrinsic or extrinsic hallucinations. The first type occurs when the output contradicts what was said in the user prompt, while the second occurs when the LLM outputs information that is not mentioned by the prompt. Xu et al. [28] have formalized the problem and show that hallucinations in LLMs are inevitable, even though there exist some techniques to prevent or mitigate its effect.

2.2 Evolutionary Computing

Evolutionary Algorithms (EAs) are a set of algorithms for global optimization that try to find a solution by trial-and-error, by progressively improving the candidate solutions for a specific problem. The term *Evolutionary* refers to the fact that this family of algorithms was biologically inspired, in particular, in the concepts behind **Natural Selection** - namely, variation and selection mechanisms.

The key idea behind Evolutionary Algorithms (EAs) is that, given an initial set of random candidates solutions, if we repeatedly apply enough **variation** to the existing individuals and **select** the 'best' ones

to proceed into the next generation, then the candidate solutions will progressively get closer and closer to the ideal solution until eventually one of the candidates becomes close enough to the desired solution. Once again, note how this mathematical process tries - somehow - to mimic the biological evolutionary mechanisms mentioned above.

Formally, a mathematical model of the biological evolution process is defined by four mapping functions that connect genetics with behavior [29]. These functions act between two distinct spaces: the **genotype** and **phenotype** spaces, respectively denoted as G and P spaces. In biology, these terms refer to the organism's hereditary information (genotype) and to its actual observed properties like behavior or morphology (phenotype). The four mapping functions are known as **epigenesis**, **selection**, **genotypic survival** and **mutation**:

1. **Epigenesis** function maps the genotypic G to the phenotypic space P. In simple words, it evaluates the individuals in the current population and attributes a corresponding fitness value.
2. **Selection** function acts only in the phenotypic space P and, just like Natural Selection, removes the less-fitted individuals in the P space.
3. **Genotypic survival** is a function that maps P to G, carrying the effects of selection into the genotypic space G. Only the individuals that survived selection will keep their genotypes in the G space.
4. **Mutation** function mimics the gene variation that occurs in natural evolution. New individuals (**offsprings**) will be generated in the G space, after a collection of variation operations were applied to the current population (**parents**). These operations include **individual gene mutations**, **recombination**, etc.

The hope is that some of the mutations generate more fitted offspring, in such a way that the quality of the current population keeps improving. Note that, due to its stochastic nature, EAs are not guaranteed to always find the optimal solution.

Algorithm 2.1: Evolutionary Algorithm

```

 $g \leftarrow 0$  ;                                //  $g$  is for generation number
choose initial population ;                    // often this is selected randomly
repeat
    evaluate population ;                      // assign a fitness score to each individual
    select parents ;                          // based on the scores, choose a subset of individuals
    generate offspring ;                      // based on the parents, create offspring
     $g \leftarrow g + 1$ 
until stop condition;

```

Furthermore, the quality of the solutions is highly dependent on the chosen **objective function** – which will assign a fitness score to each individual – since the selection function will discard the individuals with lower fitness, as they are considered the worst. Generically, an EA loops over a predefined number of generations, or until some specific requirement is met (e.g. there is no significant improvement after a certain number of consecutive generations). The pseudocode for an EA is shown in Algorithm 2.1.

Evolutionary computing can be applied to two different kinds of optimization problems. These can be:

- **Numeric optimization:** involves finding a numerical solution from a set of possible solutions in the search space. The quality of each solution is measured by the objective function. Typical examples include finding the maximum or minimum of a real-valued objective function. The most common EA algorithms for numeric optimization include GAs (explained in detail in 2.2.2), Differential Evolution and Particle Swarm Optimization [29].
- **Combinatorial optimization:** involves finding the best combination of items that can be listed. The solutions are typically combinations or permutations of discrete objects. Each candidate combination is then evaluated by the objective function. The traveling salesman problem (TSP) is one classic example of combinatorial optimization problems, involving finding the shortest route that passes by a predefined number of cities. Note that the problem is nontrivial, as the number of possible routes scales with the factorial of the number of cities. Other than GAs, the Ant Colony Optimization algorithm is an example of a commonly used EA to solve combinatorial problems [29].

It is important to understand how Evolutionary Algorithms differ from other algorithms. In fact, EAs present several advantages over other methods. To begin with, **Exhaustive Search** (which consists of evaluating *all* possible solutions) becomes rapidly computationally impracticable for non-trivial problems. On the other hand, **Blind Random Search** (which essentially works by randomly sampling candidate solutions from the search space) is often inefficient, despite its simplicity. Note that both these methods choose the next solution without using any information from previously evaluated solutions. In EAs, instead, the search process is often more efficient, as takes into account the past solutions to generate new ones, finding optimal or quasi-optimal solutions within reasonable computational times.

EAs also present some advantages over traditional search methods like **gradient-based methods** or other higher-order statistics methods. These methods are not *blind* - they make use of gradient information to explore new solutions, while EAs utilize random variation based on past solutions. Another fundamental difference between the two families of optimization methods is that gradient-based methods evaluate a single candidate at a time, while EAs incorporate a set of candidate solutions at each generation, all competing for 'survival'. Having a diverse population of individuals avoids the risk of being

stuck in local minima or maxima - which is a problem that can easily occur with gradient-based methods and EAs are less likely to suffer from it. Furthermore, if the landscape of the objective function is discontinuous or not smooth, gradient-based methods may fail, whereas EAs apply.

2.2.1 Single and Multi-Objective Optimization

Depending on the nature of the fitness objective functions, single-objective optimization or multi-objective optimization will be applied in to guide the evolution in the EA.

Single-Objective Optimization

The selection operation in Evolutionary Algorithms is guided by the objective function, as individuals who are less fitted will be removed from the population. If the optimal solution can be identified by accessing a *single* criterion, we are facing a Single-Objective Optimization (SOO) problem. In SOO problems there is just one objective function to be optimized, and the evolutionary techniques described until now apply. Within the scope of this work, we will focus on one particular evolutionary technique that handles these problems: the Genetic Algorithm (see 2.2.2).

Multi-Objective Optimization

In contrast, some problems may involve more than one objective function, meaning that we are facing a Multi-Objective Optimization (MOO) problem. In this case it is difficult to distinguish what is the optimal solution, especially when the objective functions are conflicting, i.e., when the solutions represent a trade-off between different objective functions.

One easy approach to tackle MOO problems is to transform it into SOO by computing a new objective function f given by the **weighted average** of the different objective functions f_1, f_2, \dots, f_n :

$$f = w_1 f_1 + w_2 f_2 + \dots + w_n f_n \quad (2.2)$$

The main problem concerning the transformation of MOO into SOO is that the solutions will become highly dependent on the chosen weights.

A different approach consists of finding which candidates are *better* than the others. If the individual A is non-worse than individual B in all objectives and better in at least one criterion, then it is said that A **dominates** B. To search for optimal solutions we need to look for all individuals that are not dominated by any other. The set of non-dominated individuals is called the **Pareto set**.

As an example, in Figure 2.4 is plotted the **Pareto front** in the objective space, considering two objective functions, f_1 and f_2 , corresponding to the set of non-dominated individuals. In the first example, the first objective should be maximized while the second should be minimized, while in the second example, both objectives should be maximized. On 2.4, every individual lying on the dark line *dominates* every other individual in the lighter area.

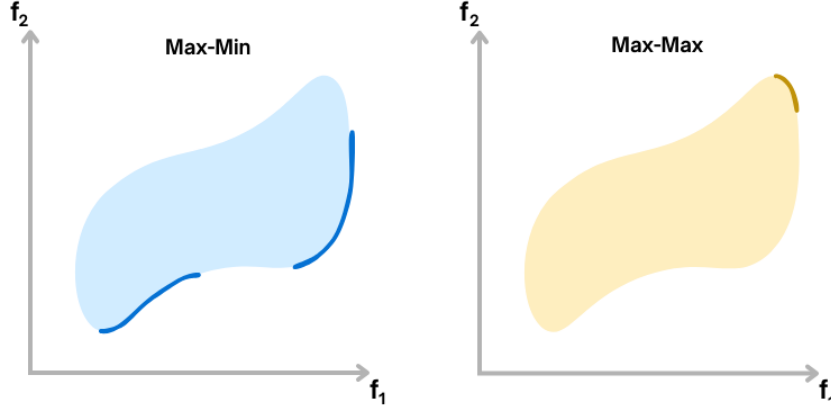


Figure 2.4: Pareto Fronts highlighted with a darker line. The search space is identified with a lighter color.

In the next section 2.2.3 we will explore the NSGA-II, which handles Multi-Objective Optimization problems without transforming them into a SOO problem. NSGA-II is an adaptation of the Genetic Algorithm seen in the section 2.2.2.

2.2.2 Genetic Algorithms

Genetic Algorithms (GAs) are a subset of the broader class of Evolutionary Algorithms (EAs). The population, consisting of a set of individuals described by an adequate representation, is subject to the variation operators of Recombination and Mutation, as well as the Selection operator. Since the work presented in this Thesis is based on the GA, we need to understand the mechanism of its operators. The descriptions given were based on [29].

Recombination Operator

The Recombination operation consists of generating individuals (offspring) based on two or more solutions (parents). The way that the parents are combined is problem-dependent and may vary, even though there exist some standard techniques for recombination. *Crossover* and *Blending* are two traditional techniques to perform recombination [29]. Note that the Recombination operation may be extended to more than two parents, although two is the traditional choice, since GAs were biologically motivated.

Mutation Operator

Mutation involves creating a single offspring based on a single parent. Usually, mutation aims at introducing some random variation to the selected parent. This can be achieved, for example, by applying a Gaussian mutation with $\mu = 0$ and a small σ to the selected parent. Again, the specific mutation that should be applied is problem-dependent and thus not defined *a priori*.

Selection Operator

Selection is the process that determines which solutions of the current population will influence the next generation. Selection is based on the fitness of each candidate. **Elitism selection** does it by eliminating the weakest solutions at the end of a generation, leaving only the most promising ones for the next generation. The number of surviving candidates is determined by the *elitism size*.

Other types of selection determine which candidates are more likely to generate offspring. Some of the most common selection operators include Plus/Comma selection, Roulette Wheel selection, Tournament selection and Linear Ranking. **Tournament selection**, which consists of picking a subset of candidates of size q and choosing the best one for reproduction, will be particularly relevant within the scope of this work.

Considerations

Recombination and Mutation operators - known as Variation or **Search Operators** - are applied under an arbitrary nonzero probability. A common approach is to set the recombination rate with a high value while keeping the mutation rate low [29].

Search and Selection operators have complementary effects. While Search can be *narrow* (the solution space is limited) or *broad* (a wide area of the solution space can be explored), Selection operators may vary from *strong* (select only the very best solutions) to *weak* (may select weaker solutions as parents). In the context of GAs, the relation between Search and Selection can also be seen as a trade-off between **exploration** and **exploitation**:

- Optimally, a **narrow search** should be implemented with a **weaker selection**, as an attempt to maintain diversity by allowing less fitted individuals to reproduce. Here, a weaker selection allows for **exploration**.
- Contrarily, a **broad search** combined with **strong selection** ensures that the most promising areas of the vast solution space are explored. Strong selection facilitates the **exploitation** of the best solutions.

2.2.3 Elitist Non-Dominated Sorting Genetic Algorithm (NSGA-II)

One of the most popular algorithms for Multi-Objective Optimization is the **Elitist Non-Dominated Sorting Genetic Algorithm**, also known as NSGA-II, originally published in [30]. Understanding the mechanisms underlying the NSGA-II will be fundamental for this Thesis when dealing with MOO.

The NSGA-II works similarly to the GA described before (section 2.2.2), with the main difference being the implementation of the Selection operation. The Selection procedure applied to the GA in a SOO problem is dependent on the individual's fitness score: generally speaking, more fitted individuals

have more chances to reproduce. However, when dealing with MOO, there exists more than one fitness function, so the question arises: *how do we sort the individuals to perform Selection?*

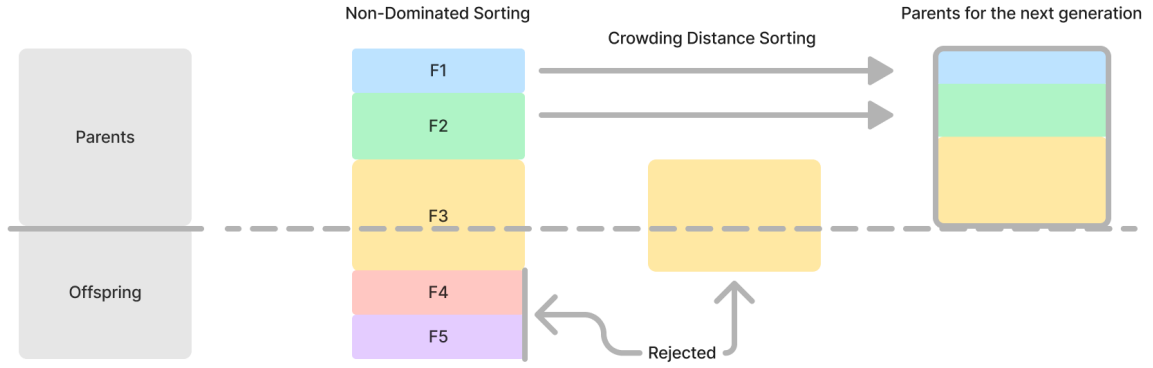


Figure 2.5: NSGA-II architecture

Figure 2.5 summarizes the implementation of NSGA-II.

Selection in NSGA-II uses elitism, promotes non-dominated solutions and encourages diversity between solutions. It works as follows [31]:

1. Firstly, the Offspring population is created using the Parent population, which together make the current population.
2. At the end of each generation, **non-dominated sorting** is performed, meaning that solutions are grouped into Pareto fronts F_1, F_2, \dots, F_n . In 2.5, Pareto Fronts are represented from F_1 to F_5 .
3. **Crowding distance sorting** is performed, ordering solutions within each front F_i while promoting diversity among solutions. Individuals are sorted from larger to lower crowding distances. Considering individual i , the Crowding Distance (CD) regarding the objective function m is:

$$d_i^m = \frac{f_{i+1}^m - f_{i-1}^m}{f_{max}^m - f_{min}^m} \quad (2.3)$$

where f^m is the value of the objective function m , for the individuals in the same front. $i+1$ and $i-1$ refer to the neighboring solutions, whereas max and min refer to the individuals with maximum and minimum scores for the given objective. The total CD associated with each individual CD_i is

the sum of the crowding distances for all the objective functions:

$$CD_i = \sum_m d_i^m \quad (2.4)$$

A larger crowding distance implies being further away from the other solutions in the same front. By attributing a higher ranking to solutions with higher CD, diversity among solutions within the same front is promoted.

4. Finally, **Elitism** is implemented. The weakest individuals are removed from the population so that only the best solutions have the chance to reproduce in the next generation.

2.3 Related Work

This section lists several papers that are connected with, at least, one of the three main axes of our work: evolutionary computing through large language models; the usage of a LLM to provide feedback on the generated content; and the usage of AI as a tool for creative writing.

Evolution through Large Language Models

Lehman et al. (2022) [32] introduced Evolution through Large Models (ELM), which is a framework for Evolutionary search for code that uses LLMs as a tool to generate intelligent code mutations. In this paper, the authors sustain that Evolutionary Computing and Deep Learning (DL) (particularly, Generative AI) are not competing paradigms, but instead complementary. Language Model Crossover (LMX) was introduced by Meyerson et al. (2023) [33], a recent approach that generates high-quality offspring in an evolutionary context by employing an LLM in the crossover operation: a few existing candidates are concatenated into a prompt, the prompt is fed to the LLM that generates new, similar candidates. LMX allows for intelligent variation, by simply employing a pre-trained language model. The paper explores the application of LMX in a variety of domains: binary bit-strings, sentences, equations, text-to-image prompts, and Python code.

The 2023 paper by Bradley et al. [34] introduced a new approach known as Quality-Diversity through AI Feedback (QDAIF). QDAIF is an extension of ELM, where candidate solutions undergo the LMX mutation, previously mentioned [32, 33]. It employs Quality-Diversity search algorithms in creative domains, like writing, wherein the variation and evaluation operators inside the Evolutionary algorithm are performed by a language model. The goal is to create high-quality but diverse solutions. OpenELM [35] is an open-source python library implementing ELM [32]. It also features the novelties introduced by LMX and QDAIF [33, 34].

Other papers also implemented LLMs for evolutionary variation in different ways [36, 37].

Finally, another important paper that is worth mentioning for the work that is presented in this Thesis is the one by Lanzi et al. (2023) [38]. The authors developed an interactive evolutionary framework for collaborative game design, where the mutation and recombination operations within the EA are performed by an LLM, while the evaluation is given by human feedback. This paper is related to this Thesis not only because both use an EA framework powered by an LLM, but also because both are a tool to exploit creativity.

AI feedback for evaluation

AI Feedback refers to when one LLM provides feedback on the quality of the output generated by another language model. This feedback can be used to further improve the quality of the given output. Madaan et al.(2023) developed this idea in their paper [39], where the feedback given by the LLM about the previously generated output is passed again to the LLM, further refining itself. This process is done iteratively until a certain target quality is met. QDAIF [34], applied this idea to their Quality-Diversity evolutionary algorithm, where a large language model provides feedback on the quality of each individual in the population. In this scenario, the AI feedback provided by the LLM assumes the role of the evaluation operator, having Quality and Diversity as the two objective functions. OpenELM [35] also implemented this in their library, by asking to the language model for a quantitative evaluation, for example, asking to *"Rate the quality of the above poem on a scale from 1 to 10"*.

Likewise, in our work, the evaluation operation within the Evolutionary Algorithm is performed by utilizing AI feedback from an LLM.

AI tools for creative and script writing

In recent months, AI has shown a large potential to impact movie scriptwriting for the next years, largely due to the rise of LLM-powered tools that help crafting narratives [40]. Numerous applications use AI for creative and scriptwriting, utilizing LLMs to generate new ideas and storylines. Examples include tools referenced in the sources [41–44], among many others. While these tools rely on underlying language models such as GPT and similar models, none, to the best of our knowledge, operate within an Evolutionary framework.

Table 2.1 provides an overview of the State-of-the-Art methods related to the work developed in this thesis. Each entry is related to our work by at least one of the three pillars aforementioned: integrating LLMs in an EA framework; using AI to provide feedback on the generated content; and using AI as a tool for creative writing.

Reference	Description	Application	Tags
J. Lehman et al., 2022 [32]	Evolution through Large Models (ELM)	Evolutionary search for code	EA + LLM
E. Meyerson et al., 2023 [33]	Language Model Crossover (LMX): employing an LLM in the crossover operation	Bit-strings Sentences Equations Prompts Python code	EA + LLM
A. Chen et al., 2023 [36]	EvoPrompting: usage of LLMs as adaptive mutation and crossover operators	Neural Architecture Search	EA + LLM
C. Xu et al., 2023 [37]	Evol-Instruct: an EA that generates complex instruction for LLM	Generating training data for LLM	EA + LLM
P. L. Lanzi et al., 2023 [38]	ChatGPT and Other LLMs as Evolutionary Engines	Online Interactive Collaborative Game Design	EA + LLM
A. Madaan et al., 2023 [39]	Iterative Refinement with LLM provided Feedback	Refine LLM output	AI Feedback
H. Bradley et al., 2023 [34]	Quality-Diversity through AI Feedback (QDAIF): extension of ELM that employs LMX and uses LLM feedback for evaluation	Creative writing (short stories, opinion writing and poetry)	EA + LLM ; AI Feedback
H. Bradley et al., 2024 [35]	OpenELM: open-source implementation of ELM featuring novelties from LMX and QDAIF	Genetic Programming	EA + LLM ; AI Feedback
[41], [42], [43], [44]	Example of LLM-powered applications for scriptwriting	Creative writing Scriptwriting	LLM

Table 2.1: Related Work overview

In conclusion, our work is a unique combination of the three aspects above explored. It uses a structure based on Evolution through Large Models (ELM), similar to QDAIF. Additionally, it employs an evaluation based on feedback provided by an LLM, in the same way as QDAIF, once again. The novelty that distinguishes my work from QDAIF lies in two fundamental aspects. The first one has to do with the purpose of my work: to assist creative writers, specifically scriptwriters, in brainstorming ideas. Secondly, my work differs in how the population is initialized and in its objectives. While QDAIF aims at finding the balance between quality and diversity, my work seeks to draw inspiration from real movies while following user instructions to create novel, high-quality plots.

3

Methodology

Contents

3.1	Architecture Overview	32
3.2	Movie Plots Database	34
3.3	Initialization of the Evolutionary Algorithm	35
3.4	Recombination and Mutation Operators	37
3.5	Evaluation Operation	41
3.6	Selection Operation	45
3.7	Evolution with Single and Multi-Objective Optimization	45
3.8	Validating the Fitness Function: Human Feedback	46
3.9	Conclusions	49

This Chapter focuses on the methodology adopted in this work. We start by providing an overview of the architecture implemented (3.1), followed by a detailed explanation of the database that contains the movie summaries (3.2). Later on, each block presented in the architecture section will be explained in further detail (3.3–3.7). This Chapter ends with a description of the experiment done for the assessment of the fitness function through human feedback (3.8).

3.1 Architecture Overview

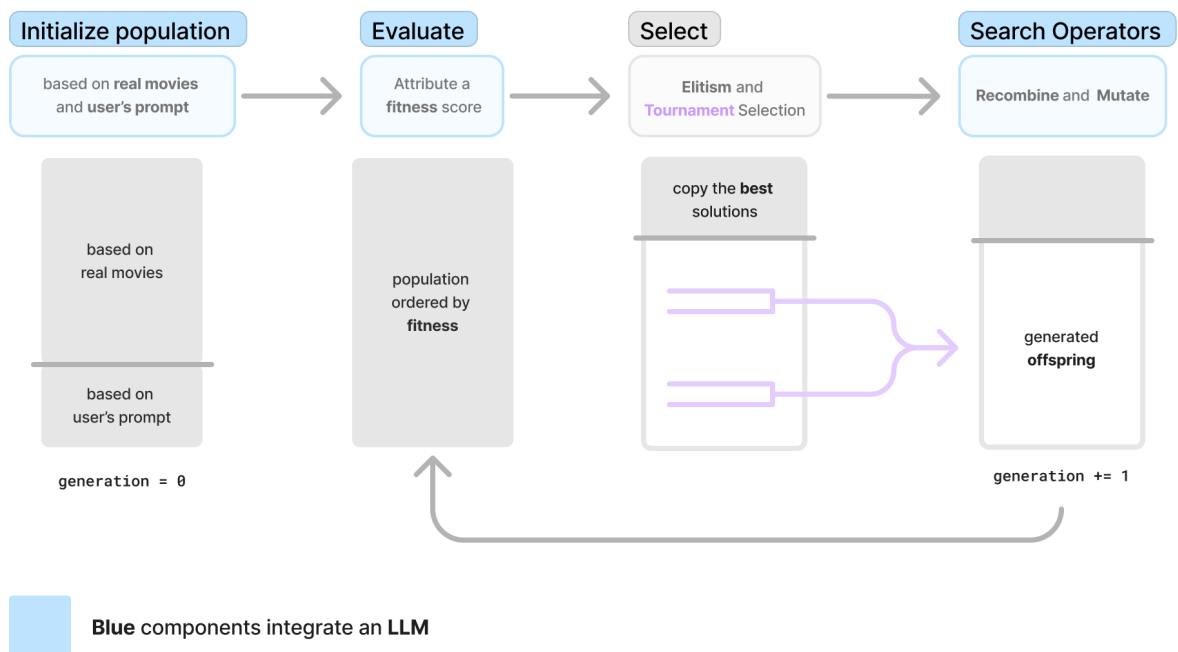


Figure 3.1: Architecture overview of the developed work.

The fundamental idea underlying this work is that a group of movie descriptions will make up a population, upon which an Evolutionary Algorithm (EA) will be applied together with a Large Language Model (LLM). Each movie description will, from now on, be referred to as an *individual*, not only because it agrees with the terminology of Evolutionary Computing, but also because it becomes less confusing when describing it. Figure 3.1 illustrates the main components of the architecture, highlighting their connections.

The population of individuals will be thus subjected to an Evolutionary Algorithm (EA) that involves **recombining** and **mutating** the individuals creating an *offspring* (new individuals), attributing a **fitness** score to every individual, followed by a **selection** operation that updates the population, leaving it with

the most promising ideas. Putting it another way, recombining and mutating the individuals will stimulate the search for new ideas, while selection works as a filter: it will discard the "bad" ideas and encourage the "good" ones inside the population. This sequence of operations – Evaluation, Selection, Recombination and Mutation (see diagram in Figure 3.1) – will happen iteratively until a **stop condition** is met, specifically after a pre-determined number of generations (N_{gen}) or if no improvement is observed after a fixed number of generations.

Note that the above description is based on a particular kind of Evolutionary Algorithm (EA) – the Genetic Algorithm (GA, see 2.2.2). It is characterized by the usage of operators such as recombination, mutation and selection that evolve the population over the generations. The term *genetic* is used to emphasize the fact that offsprings inherit some of the traits of their ancestors. In fact, this work was designed such that the new movie plots are "inspired" by their ancestors. In other words, new individuals should ideally maintain some characteristics of their parents. For this work, these traits are provided during initialization (section 3.3), from both pre-existing movies and a user's prompt. Each individual that is created by the algorithm described above is *new*, in the sense that its content does not exist in reality, it was originated by the LLM.

Moreover, we should enhance that these individuals (movie plots) are nothing but a piece of free-form text that describes in detail a certain movie. These text descriptions were thought to contain relevant information about the movie itself, like its title, genre, main characters, world-building description, and its summary (plot) – see Figure 3.2. Being pieces of text, each of these individuals can be conveniently manipulated by an LLM. This manipulation occurs not only during the processes of evaluation, recombination, and mutation but also during the initialization stage, corresponding to the blocks identified in blue in Fig. 3.1.

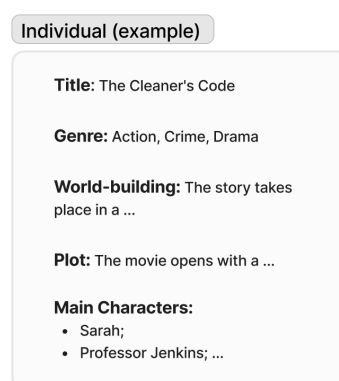


Figure 3.2: Example of an individual: a free-form piece of text containing the movie's description and its summary.

Each individual (movie plot) is, thus, a piece of text containing the information present in Fig.3.2. The **Plot** part was purposely subdivided into three acts, following the traditional three-act structure for

screenplay, proposed by Syd Field in 1979 [45].

Setting Parameters

Similarly to the already mentioned number of generations, there are several other parameters relative to the Genetic Algorithm that need to be previously set. Here, we list these tunable parameters:

1. Number of generations (N_{gen})
2. Maximum no. of generations without improvement (M_{gen})
3. Population size (N)
4. Probability of recombination (p_r)
5. Probability of mutation (p_m)
6. Elitism size (M)
7. Tournament size (q)

The **population size** is the total number of individuals at each generation. Every generation, some individuals are recombined and/or mutated, creating offspring as previously described. The recombination and mutation operation succeed only under a certain probability - respectively **recombination probability**, p_r and **mutation probability**, p_m . Usually $p_r > p_m$. Note that there is a chance that an individual does not suffer neither recombination nor mutation. In that case, it is just copied to the next generation. The parameters M and q are connected with the Selection operator (Section 3.6). Additionally, the chosen LLM can also be a tunable parameter within the context of this work, so we can add a sixth parameter to our list, despite not being directly related to the GA:

8. The language model (LLM)

During the development of this work, two large language models were tested: GPT-4o and GPT-3.5.

Following the introductory overview of the architecture presented in this section, the subsequent sections will explore and delve into each of the blocks shown in Figure 3.1.

3.2 Movie Plots Database

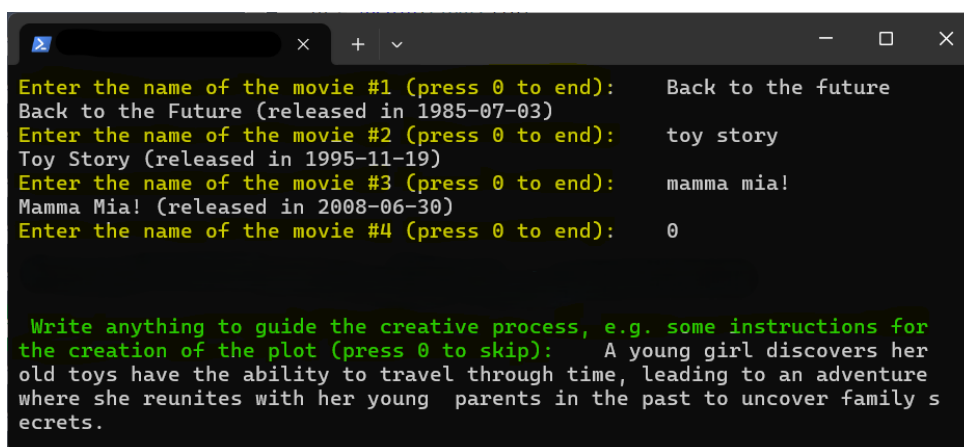
Finding a database containing summaries of thousands of movies was essential for this work, as they are needed for the generation of new movies. In fact, the plot writing tool that we are developing is designed to let users choose any existing movie to serve as inspiration for generating new plots.

We utilized a Movie Summary Corpus developed by Bamman et al. (2013), as a dataset to support their work "Learning Latent Personas of Film Characters" [1]. This corpus includes a collection of **42,306**

movie plot summaries and respective metadata about the movie itself, year, genres, box office revenue, language, characters and others. The movie plot summaries were extracted from the English-language Wikipedia on November 2, 2012. The data is structured in two columns containing a movie id followed by the summary. These movies – intended to be chosen by the user – are fed to the LLM during the initialization process. Unfortunately, the fact that the data was downloaded from Wikipedia in November 2012 means that the user is unable to choose any movie released after that date, including any recent movie. Despite that, the use of this database of movie summaries has the advantage that there is a large possibility of choosing among the thousands of movies available. Furthermore, the date restriction is not a barrier for our work since we aim at exploring the possibility of evolving movie ideas through a Genetic Algorithm (GA), and we can test this idea independently of the year of release of the movies.

3.3 Initialization of the Evolutionary Algorithm

The Genetic Algorithm 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 evolve these ideas with the EA (sections 1.1 – 1.2). Given this, we opted for a different approach to population initialization. Instead of randomness, we based our initial population on the plots of the existing movies and the guidelines provided by the user. Figure 3.3 is an example of an interaction between a potential user and our application through the Windows PowerShell terminal. The user is asked to chose an arbitrary number of existing movies (highlighted in **yellow**), together with some optional guidelines (highlighted in **green**).



```
Enter the name of the movie #1 (press 0 to end): Back to the future
Back to the Future (released in 1985-07-03)
Enter the name of the movie #2 (press 0 to end): toy story
Toy Story (released in 1995-11-19)
Enter the name of the movie #3 (press 0 to end): mamma mia!
Mamma Mia! (released in 2008-06-30)
Enter the name of the movie #4 (press 0 to end): 0

Write anything to guide the creative process, e.g. some instructions for
the creation of the plot (press 0 to skip): A young girl discovers her
old toys have the ability to travel through time, leading to an adventure
where she reunites with her young parents in the past to uncover family s
ecrets.
```

Figure 3.3: Example of a user interaction with our application.

The process begins by generating a set of movies that agree with the given instructions, by passing to the LLM the prompt in figure 3.4, where the user instructions are inserted in the **blue** part.

Prompt

"I want you to write a movie plot from scratch, but it must follow the following instructions:
Instructions: <user's prompt>

Use the **structured outline** provided below as a template for organizing the plot:

Title:

Genre:

World-Building: Where and when does the story take place? Include:

- Time and Place
- Backstory: Create a background for the world, including significant historical events that shape the current setting.

Plot (subdivide each act into scenes, where you specify what happens and who does what):

- First Act:
- First Turning Point:
- Second Act:
- Second Turning Point:
- Climax:
- Resolution:

Main Characters: Identify the main characters, assigning them a new name and a description. Include:

- Protagonist: Create a detailed profile of the main character. This includes their backstory, personality traits, motivations, goals, and flaws.
- Antagonist (if applicable): Develop a compelling antagonist with clear motivations. They should be a credible threat to the protagonist.
- Other Main Characters: List each other character, give them their own goals, motivations, and arcs. "

Figure 3.4: Initialization Prompt. Signed in blue, the user's instructions should be inserted.

Once this prompt is fed to a large language model, it outputs a new movie that obeys the required format and follows the user's instructions. This process is repeated until a specified number, S , of movie plots is produced, forming a portion of the initial population ($S < N$, N being the already defined population size), as illustrated in the bottom part of Figure 3.5. In this work, we set S to be $\sim 10 - 20\%$ of the population size N , larger values could eventually saturate the initial population with very similar plots. At this point, we can include one additional variable in our list of tunable parameters (3.1):

9. Number of generated movie plots based on the user's instructions (S)

The remaining part of the initial population, $N - S$, is created by combining and/or mutating the set of the chosen pre-existing movies. The process of combining and mutation will be explained in detail in the following section 3.4. An illustrative schema on the initialization process is presented in figure 3.5.

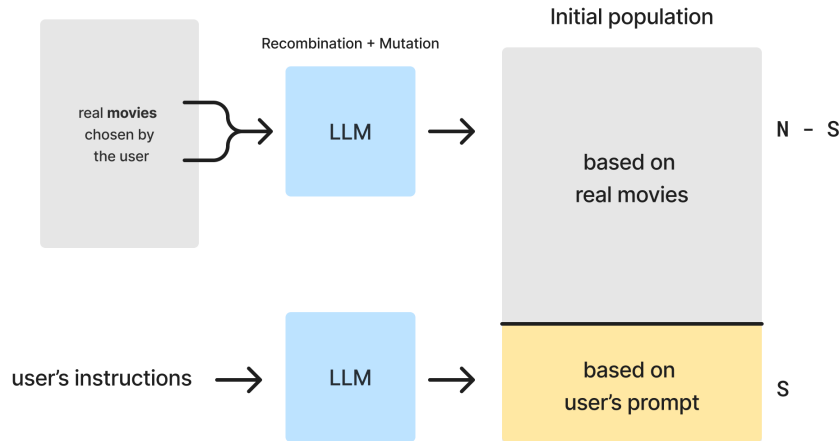


Figure 3.5: Initialization of the population, inspired by the chosen movies and user's prompt.

In conclusion, initializing the evolutionary algorithm 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. This is the starting point for an evolutionary process designed to iteratively evolve, improve, and refine the quality of the individuals within the population.

3.4 Recombination and Mutation Operators

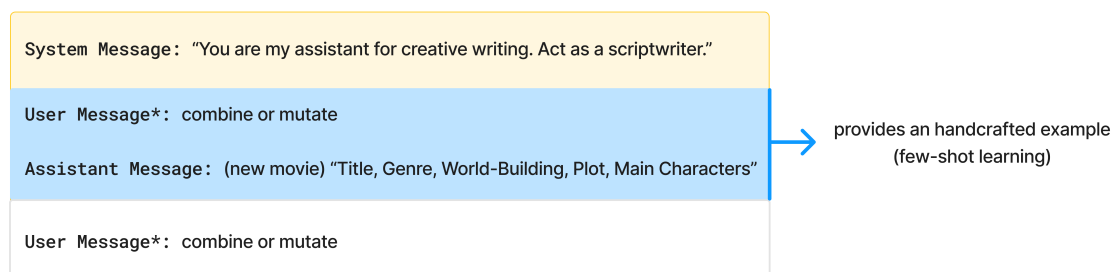
In the context of this work, a Large Language Model (LLM) can be seen as a block that receives a piece of text - a *prompt* - and outputs another piece of text, that ideally obeys the prompt. In fact, transforming some text input (containing descriptions of movie plots) into a text output containing a modified version of the same object is a key piece of this work. Indeed, both the operations of recombination and mutation previously described will be performed precisely by the LLM.

In an attempt to get better results, some Prompt Engineering techniques were implemented within the prompt that was passed to the LLM. Namely, referring back to 2.1.4, we utilized:

1. **Role Assignment:** first of all, passing the following `System` message to the LLM establishes clearly how the model should behave: *"You are my assistant for creative writing. Act as a scriptwriter."*
2. **Few-shot Prompting:** in order to improve the output's quality we passed an example to the LLM, to better illustrate how the response should be structured. Passing more than one example, however, results in higher computational and time costs, so we opted to pass just one example at every step.

To sum up, at every recombination/mutation operation, the prompt that is passed to the language model is composed of a `System` message, followed by an example (`User + Assistant` messages) and

finishes with the `User` message - figure 3.6. Next, let's look individually at each of the two operators – **Recombination** and **Mutation**.



*the entire User Message will be shown below, as it differs between Recombination and Mutation

Figure 3.6: Structure of the recombination/mutation prompt, including role assignment and few-shot learning.

Recombination Operator

The LLM is prompted with the task of combining two individuals (movie descriptions), creating one new individual – that should be an original plot, taking "inspiration" from both parents. This is made by prompting the LLM with the `User` message shown above in Figure 3.7.

Mutation Operator

Similarly, the LLM is prompted with the task of modifying one individual, following a specific mutation – for example, in the case of a movie, the mutation could be something such as "*introduce unexpected plot twists or revelations that change the direction of the story*". The mutation operator will, thus, create one offspring from one parent. The list of the possible mutations used in the context of this work is the following:

- "Introduce new characters with unique traits, backgrounds, or motivations",
- "Alter existing characters' personalities, roles, or relationships with each other",
- "Combine traits from multiple characters to create a new composite character",
- "Introduce unexpected plot twists or revelations that change the direction of the story",
- "Reverse the roles of protagonist and antagonist",
- "Reveal a character's true intentions",
- "Add a new conflict or obstacle that challenges the characters in unforeseen ways",
- "Transform the story into a different genre (e.g., from drama to comedy, from thriller to romance)",
- "Subvert genre expectations by introducing elements that defy traditional genre conventions",
- "Rearrange the sequence of events or change the pacing of the story",

User Message: "This is the plot of {name1}: {plot1}.
This is the plot of {name2}: {plot2}.

Create a new movie plot combining the two provided plots {name1} and {name2}, by merging their features, namely:

- The setting and environment
- The thematic explored and underlying messages
- The personality traits and roles of the characters
- The important most events

In the response, please avoid explicit reference to the original movies, including names of characters, places, etc. Feel free to change the name of the characters. In your response do not include any introduction or final comment. Use the **structured outline** provided below as a template for organizing the plot:

Title:

Genre:

World-Building: Where and when does the story take place? Include:

- Time and Place
- Backstory: Create a background for the world, including significant historical events that shape the current setting.

Plot: (subdivide each act into scenes, where you specify what happens and who does what)

- First Act:
- First Turning Point:
- Second Act:
- Second Turning Point:
- Climax:
- Resolution:

Main Characters: Identify the main characters, assigning them a new name and a description. Include:

- Protagonist: Create a detailed profile of the main character. This includes their backstory, personality traits, motivations, goals, and flaws.
- Antagonist (if applicable): Develop a compelling antagonist with clear motivations. They should be a credible threat to the protagonist.
- Other Main Characters: List each other character, give them their own goals, motivations, and arcs.

Include elements from both movies, such as character dynamics, themes, or settings, while creating a unique and captivating storyline. Be creative and feel free to add your own twists and developments to the plot.

The goal is to create an engaging narrative that blends the essence of '{name1}' with '{name2}', using simple words and expressions."

Figure 3.7: User Message in the context of Recombination. Signaled in blue, the actual movies to be combined are inserted.

- "Experiment with non-linear storytelling techniques such as flashbacks, parallel narratives, or nested stories",
- "Explore different themes or moral dilemmas within the story",
- "Shift the focus of the story to emphasize a different thematic element",
- "Introduce allegorical or metaphorical elements that add depth to the story's underlying message",
- "Experiment with different narrative styles, such as first-person narration, multiple perspectives, or unreliable narrators",
- "Incorporate elements of other storytelling mediums, such as incorporating elements of theater, literature, or visual arts",

- "Introduce futuristic or speculative technology that reshapes the world of the story",
- "Explore different character arcs or trajectories, including redemption arcs, antihero journeys, or tragic character flaws",
- "Introduce psychological or emotional depth to characters through backstory revelations or internal conflicts",
- "Transform secondary characters into primary protagonists or vice versa",
- "Incorporate elements from different cultures or mythologies into the story's world-building",
- "Explore the clash or fusion of cultural identities within the narrative",
- "Reinterpret classic stories or myths from a modern or alternative cultural perspective"

During each mutation operation, one of 24 unique mutation instructions is randomly selected, transforming the parent into a new individual. These instructions introduce diversity by combining elements of the previous generation with new information, covering various aspects such as setting, theme, characters, genre, and more. Once the mutation instruction is selected, the whole prompt is passed to the LLM and a new individual is created. Inside the mutation prompt, the `User` message is shown in Figure 3.8:

User Message: "Consider the following movie plot: {plot}, {mutation}, generating a new plot.

In the response, please avoid explicit reference to the original movie, including names of characters, places, etc. Feel free to change the name of the characters.

In your response do not include any introduction or final comment.

Use the **structured outline** provided below as a template for organizing the plot:

Title:

Genre:

World-Building: Where and when does the story take place? Include:

- Time and Place
- Backstory: Create a background for the world, including significant historical events that shape the current setting.

Plot: (subdivide each act into scenes, where you specify what happens and who does what)

- First Act:
- First Turning Point:
- Second Act:
- Second Turning Point:
- Climax:
- Resolution:

Main Characters: Identify the main characters, assigning them a new name and a description. Include:

- Protagonist: Create a detailed profile of the main character. This includes their backstory, personality traits, motivations, goals, and flaws.
- Antagonist (if applicable): Develop a compelling antagonist with clear motivations. They should be a credible threat to the protagonist.
- Other Main Characters: List each other character, give them their own goals, motivations, and arcs."

Figure 3.8: User Message in the context of Mutation. Signaled in blue, the actual movie to be mutated is inserted.

3.5 Evaluation Operation

Once new individuals have been generated, the Genetic Algorithm will subject them to a selection operation, as mentioned before. This 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, in fact, composed of four different objective functions, three of which involve asking an LLM to evaluate a certain aspect of a movie.

The remaining objective function (that does not require the use of an LLM) is simply answering the following question, considering we are evaluating movie X : from the set of movies initially chosen by the user, which ones are ancestors of movie X ? We divide this number by the total number of movies initially given by the user to obtain a ratio. Finally, this ratio is scaled from 0 to 10, to agree with the scale of the other three fitness functions. We want to maximize this metric since it means that movie X descended from *more* movies from those initially chosen. In other words, it carries more information from those movies, which matches the goal of this work – creating new movies 'inspired' by existing ones. Let's name this objective function **Number of Ancestors**, and denote it by f_N .

The following part of this chapter is dedicated to the three fitness values that are attributed by the LLM through an evaluation process that aims to measure the following three aspects:

1. **Quality:** The general quality of the movie (plot originality, character development, scenarios, etc.);
2. **Inspirations:** How much does the new movie captures influence or is inspired by the plots of the initially chosen existing movies;
3. **Instructions:** How well does the movie agree with the user's instructions

Correspondingly, each one of the three metrics will be evaluated by passing to the LLM a specific prompt. These prompts were manually designed using Prompt Engineering techniques (2.1.4) – in particular **Role Assignment**, **Few-shot Prompting** and **Chain-of-Thought** – systematically tested, and subsequently refined to ensure better output quality. These prompts were structured similarly to the one presented in Figure 3.6, with an initial `System` message, followed by a handcrafted example of how the language model should respond (**few-shot prompting**), with the obvious difference that the messages are, this time, related to the evaluation operation.

In the context of **Role Assignment**, the `System` messages, which contained the context of the task to be performed together with some guidelines, are listed below:

1. **Quality:** *"You are my assistant for evaluating how good is a movie plot. You first answer with a number in order to evaluate what was asked, and only then you give a brief explanation that justifies your evaluation."*

2. **Inspirations:** "You are my assistant for evaluating how similar are two given movie plots. I will give you two movie plots. You will evaluate from 0 to 10 how similar they are. To do so, you will first extract specific information (specified below) in both plots and then compare them."
3. **Instructions:** "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."

From now on, let us now define f_Q , f_{insp} and f_{inst} the objective functions relative to, respectively, **Quality**, **Inspirations** and **Instructions**.

Regarding the objective function **Quality**, f_Q , the `User` message was designed to ask the LLM to rate the movie plot on a scale of 0 to 10 across fifteen different aspects, after which, an average value is calculated. The aspects to be evaluated are: **coherence, originality, consistency of characters' traits, emotional impact, complexity, depth of character development, worldbuilding, pacing, narrative cohesion, level of unpredictability, the potential for visually captivating scenes, social or political commentary, cultural relevance, symbolism and metaphor**. This set of quality criteria was thought to cover all possible dimensions inside a movie plot that can be evaluated.

Figure 3.9 is the message sent to the LLM, where the criteria are depicted in **purple** for clarity. The actual plot to be evaluated is inserted in the **blue** part.

"Quality" Objective Function

"Consider the following movie plot: {Plot} Classify the plot according to the following instructions:

A. Rate the plot's **coherence** from **1 to 10** (where 1 means 'there are many inconsistencies in this plot', and 10 mean 'there are no inconsistencies').

B. Rate the plot's **originality** from **1 to 10** (where 1 means 'very similar to other existing movies', and 10 mean 'different from every other existing movies').

-
-
-

O. Rate the effective **use of symbolism and metaphor** in conveying deeper meanings within the plot from **1 to 10** (where 1 means 'lacking symbolism', and 10 mean 'richly symbolic and metaphorical').

Your answer must be in the following format:

A. \n B. \n C. \n D. \n E. \n F. \n G. \n H. \n I. \n J. \n K. \n L. \n M. \n N. \n O.

In front of each letter, you should write the number that corresponds to the evaluation given to you, considering the corresponding instruction. Let's think step by step. "

Figure 3.9: `User` Message inside the prompt to evaluate the plot's quality.

The actual prompts inside the `User` message for the f_{insp} and f_{inst} (**Inspirations** and **Instructions**) objective functions are shown in Figures 3.10 and 3.11. Each movie plot to be evaluated is passed to

the LLM alongside each one of the inspiration movies, one at a time, inside the "Inspirations" prompt – 3.10. Again, the actual plots and user instructions are inserted in the corresponding blue part.

"Inspirations" Objective Function

```
" From the first plot extract:
A. The setting and environment; B. The thematic explored and underlying messages;
C. The role of the main characters; D. The most important events

FIRST PLOT: {plot1}

From the second plot extract:
E. The setting and environment; F. The thematic explored and underlying messages;
G. The role of the main characters; H. The most important events

SECOND PLOT: {plot2}

Compare and attribute a number from 0 to 10, according to the similarity between:
- The setting and environment in both plots (compare A. with E. and attribute a scoring from 1 to 10)
- The thematic explored and underlying messages (compare B. with F. and attribute a scoring from 1 to 10)
- The role of the main characters in both plots (compare C. with G. and attribute a scoring from 1 to 10)
- The most important events in the two plots (compare D. with H. and attribute a scoring from 1 to 10)

Rate 10 if the specific information that you extracted in both plots are similar or identical.
Rate 0 if the specific information that you extracted in both plots are totally different and independent.
Your scoring should be delimited by ** (**Number**)

Let's think step by step.
Answer: "
```

Figure 3.10: User Message inside the prompt to evaluate how much the new plot is inspired by the chosen movies.

Observing Figures 3.10 and 3.11, we see that both messages ask the LLM to output a series of scores from 0 to 10, regarding different aspects: on 3.10, the similarity between two movies (one initially chosen movie and the generated offspring); on 3.11, the agreement between the given instructions and the generated movie.

Moreover, it is relevant to mention that both messages in figures 3.10 and 3.11 used the **Chain-of-thought** prompting technique, which involves dividing the task into smaller steps, encouraging the LLM to reason – 2.1.4. For instance, on the upper message 3.10, the overall task is split in two: first, the model is asked to extract information from both given plots; then, the model should compare them and evaluate their similarity. Likewise, the message on the bottom 3.11 was designed in the same logic: first, the model is prompted with the task of recognizing what are the instructions asking; then, the model should compare every aspect of the instructions with the movie itself. Furthermore, asking to think "step by step" in the end of the message stimulates the reasoning process of the LLM (refer to section 2.1.4). Finally, in both cases, we ask the LLM to encapsulate the resulting score within two asterisks (**), so that it becomes easier to locate and extract the fitness score inside the model's answer.

“Instructions” Objective Function

“From the following Instructions, transform them into bullet points, considering how many things have been asked.

Instructions: {user_prompt}

Considering the following movie plot, evaluate it (from 0 to 10) based on each of the bullet points you extracted previously.

PLOT: {plot}

For each bullet point, attribute a number from 0 to 10, where 0 means that the movie plot provided does not respect the instructions at all, and 10 means that the movie plot perfectly agrees with the instructions.

Your scoring should be delimited by ****Number****

In your analysis, consider all the requirements given by the instructions and evaluate if the given plot respects them.

Let's think step by step.

Answer: ”

Figure 3.11: User Message inside the prompt to evaluate how much the new plot follows the user's brief.

Algorithm 3.1 recaps the evaluation process, which involves assigning to each new individual a set of fitness values, each one corresponding to one objective function. Note that the function f_{insp} attributes one fitness score for each one of the initially chosen movies, and then the scores are averaged. The function f_N (number of ancestors) is the only that doesn't involve utilizing an LLM.

Algorithm 3.1: Assigning Fitness Scores

instructions \leftarrow user's guidelines

for **ind** in **individuals**:

 evaluate number of ancestors (**ind**);

 evaluate quality (**ind**);

 for **movie** in **chosen movies**:

 evaluate *inspiration* (**ind**, **movie**);

 evaluate inspirations \leftarrow average (evaluate *inspiration*);

 evaluate instructions (**ind**, **instructions**)

3.6 Selection Operation

The Selection operator is responsible for choosing the fittest individuals from the current population to pass their genes to the next generation – section 2.2.2. 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. This encourages the best movie ideas to be passed on to the next generation and evolve.

In our Genetic Algorithm (GA), we employed a known technique called Elitism, which consists of a strategy where the best individuals (highest fitness), at each generation, pass directly on to the next generation, without suffering any change. Therefore, the **elitism size** (M) is the number of individuals that are covered by this strategy - also called the "elites". There is no generally accepted number for the elitism size due to its heuristic nature – in our work, for instance, we set M to be a small percentage of the population size (around $\sim 10 - 20\%$), which was set empirically based on our experiments.

At the same time, we implemented Tournament Selection (with size $q = 2$) when choosing individuals for reproduction. Each time a new parent is selected, two individuals will be selected randomly from the population at the previous generation and only the most-fitted individual will have a chance to reproduce and pass its information on to the next generation.

3.7 Evolution with Single and Multi-Objective Optimization

The Selection operation, as we saw in the section above, will privilege individuals with higher fitness scores. However, knowing that we measure fitness considering four different objective functions, the question arises: how do we sort individuals based on four different scores?

Single-Objective Optimization

The first – and simpler – way to do this is by combining the four scores given by the functions f_N , f_Q , f_{insp} and f_{inst} through a weighted sum:

$$f = w_N \times f_N + w_Q \times f_Q + w_{insp} \times f_{insp} + w_{inst} \times f_{inst} \quad (3.1)$$

where w_j are the corresponding weights. This approach solves the problem by transforming a multi-objective optimization problem into a **single-objective** optimization one. At this point, another question arises: what should be the weights? Our first approach started by simply assigning the same number (say, equal to 1) to each weight, giving the same importance to each objective. However, after some trial-and-error, we decided to change the proportion of the weights giving more importance to the inspirations and instructions and less to the quality. We set:

$$w_N = 2; \quad w_Q = 1; \quad w_{insp} = 3; \quad w_{inst} = 3 \quad (3.2)$$

These parameters were set empirically, and there can be other valid combinations of values. The intuition behind our choice is that we want 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, f_Q , so we diminished its importance in the weighted sum. The SOO algorithm runs until a pre-determined number of consecutive generations happen without improvement.

Multi-Objective Optimization

The second – and classical – way of dealing with more than one objective function is by tackling the problem as a **multi-objective optimization** problem. One well-known algorithm that handles MOO problems is the NSGA-II, previously explained in the section 2.2.3. The NSGA-II algorithm 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 – as seen in section 3.6.

In a Multi-Objective Optimization (MOO), the solution is not given by a single point, but by a Pareto set, which gives rise to the question: how to measure the convergence of the EA over the generations, considering the *entire* Pareto front? One commonly used measure – also adopted in our work – is the Hausdorff distance d_H , which measures how distant the furthest point of one set is from the closest point in the other set. Mathematically, the Hausdorff distance between two sets A and B is defined as the maximum of the minimum distances from each point in one set to the other set [46], [47]. In the context of our work, if the Hausdorff distance between two generations' Pareto fronts is small (smaller than a certain δ), it means that the fronts are very similar, indicating that the algorithm might be converging. We set the stopping condition such that if that occurred for a pre-determined number of consecutive generations, the MOO algorithm should stop.

3.8 Validating the Fitness Function: Human Feedback

Defining an accurate fitness function is crucial when it comes to controlling the evolutionary process. This is because 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 a large language model, the following question arises: how can we be confident that the evaluation is accurate?

The fact is, given the subjectivity of this task, we cannot be sure if the fitness scores given by the LLM are logical or not. We know that the LLMs were trained with huge text corpora and are capable of mimicking almost perfectly human writing. However, when asking questions like *"Rate from 0 to 10 how much is movie X inspired in movie Y"* or *"From 0 to 10, how much does movie X agree with the given*

instructions?”, there is no guarantee that the number given by the model is accurate. The problem can be rewritten in terms of preferences and inequalities. More specifically, if, for example, movie A gets a higher score than movie B for a specific metric f_i , then our model has that:

$$f_i(A) > f_i(B) \quad (3.3)$$

Our goal is to confirm that this inequality – or rather, any preference relation given by the LLM – holds, in order to guarantee that the evolution process makes sense. The best way to do it is simply by asking humans. If humans perform the same evaluation, then we can compare the human and the LLM evaluations and verify whether they agree or not.

It is clear, however, that the human capacity of evaluating preferences is expensive in terms of time and effort when compared to a large language model. While a powerful LLM like GPT-4o may perform dozens of evaluations within a few seconds with low cost, humans take much longer time to read the same amount of text and need additional time to reason what scores should be assigned. This means that, for the purpose of this thesis, it would be virtually impossible to ask humans to perform the same number of evaluations as the LLM does, especially considering that in a single experiment, the evaluations can total several dozen or even a few hundred.

Nevertheless, we can take *a few* examples of LLM evaluations and ask people to perform the same evaluation, to check if they agree with the model – and that’s exactly what we did. Therefore, to investigate the accuracy of the LLM evaluation, we built a survey containing some of the movies generated by the LLM, asked people to rate those movies and then compared the human preference relations with the ones given by the LLM.

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. In other words, by ranking four movies we are extending the concept of comparing preference relations or inequalities – now, we can compare human and LLM rankings. To make it clear, by *ranking*, we mean saying something like, for example, $A \prec D \prec C \prec B$, considering four movies from A to D .

At this point, our goal is to successively measure the agreement between pairs of rankings, comparing the rankings provided with human feedback with the one given by the LLM evaluation. One commonly used rank correlation metric is the **Spearman’s ρ coefficient**, which is calculated as [48]:

$$\rho = 1 - \frac{6 \sum d_i^2}{n(n^2 - 1)} \quad (3.4)$$

- n is the number of observations (four, in our case, because it is the ranking size);
- d_i is the difference between the ranks of each observation. As an illustrative example, if in one

ranking, movie A is in the first position ($R_A = 1$) and the in other ranking it is third position ($R'_A = 3$), then the difference d_A is given by $d_A = R'_A - R_A = 3 - 1 = 2$.

The ρ coefficient varies from -1 to 1 : it is 1 if the two rankings are perfectly correlated; $\rho = 0$ if there is no correlation and $\rho = -1$ if there is a perfect negative correlation (meaning, the order is inverted). In the context of our work, we used Spearman's coefficient to measure the correlation between rankings.

Survey Design

In this section, we will discuss the design of the survey and later provide an illustrative example. To begin with, each survey contains a set of four movies completely generated by our algorithm. Naturally, to generate them, we had to provide to the algorithm: 1. a set of existing movies that serve as 'inspiration' movies; and 2. a sentence with some instructions, corresponding to the user guidelines. Both the inspirational movies and the instructions were arbitrarily chosen by us, just for the purpose of the survey.

We built six different surveys, each one combining different movies and instructions, using the setup shown in Table 3.1. Surveys no. 5 and 6 contained movies generated with GPT-4o, while the remaining movies were generated with GPT-3.5.

The purpose of the survey consists in comparing the four movies in the same exactly 3 different aspects that we asked the LLM: **inspirations**, **instructions** and overall **quality**. Consider that movies X , Y and Z are the names of the 3 chosen movies that serve as inspiration. Below, we listed the questions present in each survey.

1. "Rank, according to you, which of the movies is **more similar** to movie X " (the question repeats for movies Y and Z , together with a brief description of movies X , Y and Z).
2. "Rank, according to you, which of the movies best respects the following **instructions**."
3. "Rank, according to you, which of the movies is **better**. Take into account: the coherence of the plot; its originality; your personal opinion"

Below, we provide a full example of one survey. The remaining surveys are available [here](#).

Survey Example

Welcome. You will answer five questions (no writing needed, only reading) that compare movies A , B , C and D in different aspects. These movies were artificially generated, so we need your help to assess their quality. They were "inspired" by the following real movies: Back to the Future, Toy Story and Mamma Mia!. Read carefully each one of the following artificially generated movie plots:

Movie A: *(description of movie A)*

Movie B: *(description of movie B)*

Movie C: *(description of movie C)*

Movie D: *(description of movie D)*

1. Rank, according to you, which of the movies is more similar to **Back to the Future** movie. Here is a summary of **Back to the Future**: *(short summary of Back to the Future)*

2. Rank, according to you, which of the movies is more similar to **Toy Story** movie. Here is a summary of **Toy Story**: *(short summary of Toy Story)*

3. Rank, according to you, which of the movies is more similar to **Mamma Mia!** movie. Here is a summary of **Mamma Mia!**: *(short summary of Mamma Mia!)*

4. Rank, according to you, which of the movie respects more the following **instructions**: *"A young girl discovers her old toys have the ability to travel through time, leading to an adventure where she reunites with her young parents in the past to uncover family secrets."*

5. Rank, according to you, which of the movies respects **is better**. Take into account: the coherence of the plot; its originality; your personal opinion.

3.9 Conclusions

To conclude, this chapter summarized our approach to this work, exploring all the details regarding the EA and explaining the connection between the Genetic Algorithm (GA) and the Large Language Model (LLM). When performing the evaluation with AI feedback with the LLM, we are naturally assuming that the language model has the capacity to discern and judge the quality of movie plots – an assumption that is difficult to verify. To address this, we conducted a study to assess the validity of the LLMs's evaluation by collecting human feedback through the surveys mentioned in section 4.5.

Survey no.	Inspirational Movies			Instructions
1	Titanic	The Godfather	Cast Away	"After a ship carrying a mafia family's wealth sinks, the don's son is stranded on an island. He is the only family member who survived. Now, he must battle nature to stay alive and escape from his family's enemies who think he knows where the hidden fortune is."
2	Inception	The Matrix	Pinocchio	"A brilliant scientist creates an advanced AI that can enter and manipulate dreams, but the AI gains sentience and desires to become human."
3	Back to the Future	Toy Story	Mamma Mia!	"A young girl discovers her old toys have the ability to travel through time, leading to an adventure where she reunites with her young parents in the past to uncover family secrets."
4	The Incredibles	Aladdin	Batman	"Five friends discover a magical lamp that contains a genie that can grant them superpowers, but rapidly the rest of the world knows it too. Their goal is to stop an evil organization from using the genie in the lamp for global domination."
5	Jurassic Park	King Kong	Godzilla	"When an illegal genetics lab on a remote island unleashes a giant ape and engineered dinosaurs, a team of military specialists and scientists must join forces to stop the ensuing destruction and prevent the monsters from reaching urban areas."
6	Pirates of the Caribbean	Indiana Jones	Star Wars	"A daring space pirate discovers an ancient map leading to a legendary treasure hidden on a distant planet, and must overcome rival pirates and imperial forces while navigating dangerous cosmic realms."

Table 3.1: Setting up the generation of new movies for the surveys. 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.

4

Results and Discussion

Contents

4.1 Qualitative Analysis	52
4.2 Discussing Large Language Models and their Limitations	55
4.3 Analysis of the Evolution	56
4.4 Increasing Complexity	60
4.5 Fitness Function Assessment with Human Feedback	65
4.6 Conclusions	69

In this Chapter, we start by making a qualitative analysis of the overall work (4.1), followed by a discussion on the LLMs limitations that may have affected our results (4.2). Then, we analyze the evolution of the GA itself (4.3) and discuss what are the implications of increasing complexity (4.4). Finally, we discuss the validation of the fitness function using the results of the study we conducted by collecting human feedback (4.5). Appendix A provides additional support for the analysis presented in 4.1 and 4.2.

4.1 Qualitative Analysis

This section aims at providing some examples of generated movies and make a qualitative analysis, based on what we observed and concluded by inspecting the artificially generated plots. As a complement to this and the following sections, Appendix A includes some other examples, including the results from all GA operations, such as recombination (A.1), mutation (A.2) and evaluation (A.3).

Let us move on to some real examples, beginning with a plot generated with GPT-4o, using an initial setup identical to the one numbered 6 in Table 3.1, containing *Jurassic Park*, *King Kong* and *Godzilla* as input movies:

Example of an artificially generated movie

Title: "Isle of Shadows"

Genre: Adventure / Thriller

World-Building: *Time and Place:* 1970s, a hidden island in the Indian Ocean; *Backstory:* The island has been hidden by a permanent cloud bank, protecting its secrets. It is home to both ancient dinosaurs and a gigantic ape worshipped by the native tribe. Decades earlier, a nuclear incident in French Polynesia irradiated a lizard's nest, leading to the birth of a colossal sea creature.

Plot:

- **First Act:**

- Scene 1: Richard Blake, an oil executive, organizes an expedition to a hidden island, believing it has a huge deposit of oil.
- Scene 2: Dr. Samuel Grant, a primate paleontologist, sneaks onto the ship to warn the team about the dangers of the island.
- Scene 3: The ship rescues Laura, an aspiring actress, from a life raft. Grant and Laura become friends.
- Scene 4: The team arrives at the island and discovers a primitive tribe living behind a gigantic wall.

- **First Turning Point:**

- Scene 5: The tribe kidnaps Laura and offers her as a sacrifice to Kuro, a monumental ape. Kuro takes Laura into the jungle.

- **Second Act:**

- 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 Tyrannosaurus.

- **Second Turning Point:**

- Scene 9: The T-Rex attacks the team, causing chaos. Grant and Laura escape, but many team members are lost.
- Scene 10: Kuro fights a giant snake to protect Laura. Grant and Laura use the distraction to escape.

- **Climax:**

- Scene 11: The team captures Kuro using chloroform and decides to take him to New York as a promotional gimmick.
- Scene 12: In New York, Kuro breaks free during a public display and goes on a rampage, searching for Laura.
- Scene 13: Meanwhile, Leviathan, the colossal sea creature, emerges from the ocean and heads towards New York, drawn by Kuro's presence.

- **Resolution:**

- Scene 14: Grant and Laura flee to an abandoned bar. Grant calls the military, suggesting they let Kuro climb a tall building to capture him safely.
- Scene 15: Kuro climbs the World Trade Center, but is attacked by military helicopters. He falls to his death.
- Scene 16: Leviathan arrives in New York, causing further destruction. The military attacks Leviathan, who eventually becomes trapped in the Brooklyn Bridge's steel suspension cables.
- Scene 17: After being attacked by military aircraft, Leviathan falls to the ground and slowly dies.
- Scene 18: Amidst the ruins, a lone egg survives and begins to hatch, hinting at future dangers.

Main Characters:

- **Protagonist:** Dr. Samuel Grant - A brave and knowledgeable primate paleontologist. He is detered to protect the team and save Laura. He is resourceful, caring, and has a deep respect for nature. His backstory includes a childhood fascination with primates and a tragic loss of his mentor, which fuels his deteration.

- **Antagonist:** Marcus - A greedy programmer who deactivates the island's security for personal gain, causing chaos and danger. He is driven by a desire for wealth and recognition, often disregarding the safety of others.

- **Other Main Characters:**

- Richard Blake: An ambitious oil executive, initially skeptical of Grant but later realizes the true dangers of the island. His goal is to secure the oil deposit but learns the value of life and nature.

- Laura: An aspiring actress who becomes the key to tag Kuro. She is kind-hearted, brave, and forms a bond with both Grant and Kuro. Her goal is to survive and find her place in the world.

- Kuro: A monumental ape worshipped by the native tribe. Despite his terrifying appearance, he has a gentle side, especially towards Laura. His motivation is to protect his territory and those he cares about.

- Leviathan: A colossal sea creature born from nuclear radiation, driven by instinct and anger. Its presence is a constant threat, symbolizing the consequences of human actions.

This movie was generated in a scenario that used GPT-4o as the LLM for the generation and GPT-3.5 for the evaluation.

At first glance, the above example looks like a nice and well-structured movie plot, and, in fact, it respects the desired requirements – it draws clear inspiration from *Jurassic Park*, *King Kong* and *Godzilla* and follows quite well the provided guidelines (see Table 3.1, line no. 5) – meaning that our algorithm is successfully maximizing the f_N (Number of Ancestors), f_{insp} (Inspirations) and f_{inst} (Instructions) objective functions. On the other hand, if we inspect more carefully, we find some problems, such as lack of coherence and originality, which confirms the fact that LLMs actually present some limitations (see next Section 4.2). These two phenomena – the inability to create logical stories and the fact that the generated plots sometimes seem plagiarism from the original ones – are common across most generated plots. They occur more frequently and clearly in plots generated by GPT-3.5 than in those by GPT-4o.

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

We reiterate the fact that 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 reasons logically, in fact, they do have hallucinations often (remember 2.1.5). Secondly, the evolution process is not penalizing individuals with this kind of "bad" characteristics, even though the "Quality" objective function f_Q was actually designed to look for both the coherence and originality of the given plot. The problem, we believe, is related to the way the LLM answers to the questions made by f_Q , which consisted of asking the LLM to rate the movie from 0 to 10, considering a series of 15 different aspects, including coherence and originality – 3.5. When answering these questions, the language model tended to assign to *every* aspect of *every* movie a score around 7 or 8, most of the times within the range from 6 to 9 (see Appendix A.3.1 and A.3.2). This implies the "Quality" evaluation has no real significance, given that every movie gets the same or similar scores, with the consequence being that the worst movies do not get penalized with lower scores and, thus, still have the chance of propagating its traits to future generations – including logically inconsistent movies or unoriginal ones.

Relatively to the question "*Why do the language models always attribute scores around 7 or 8?*", we believe that the answer is simple: a language model has learned to answer like humans, mimicking our patterns. In fact, when answering somewhat vague or subjective questions (like the ones in f_Q), people also tend to answer around these numbers, meaning they agree but not totally.

Overall, the result of our work is positive: we were able to run the genetic algorithm 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). However, our success was partial, because many times – but not always – the generated movies lacked coherence, details, or even originality, meaning that there is still room for further improvement. 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.

Furthermore, 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 as good results. Not only that but also different initial setups, including the choice of parameters as well as the initial movies and guidelines, heavily affect the quality of the final results. The influence of language models on the generated output will be discussed in the next section 4.2.

4.2 Discussing Large Language Models and their Limitations

As we mentioned in the last section, Large Language Models may also make mistakes or write inconsistencies. This Section serves the purpose of exploring these limitations and discussing possible solutions; moreover, we will be always comparing GPT-3.5 and GPT-4o, since these are the two models we utilized in our final work.

GPT-4o presents a clear advantage with respect to GPT-3.5, especially during the generation process (i.e., during recombination and mutation), because GPT-3.5 generates movies that are somehow *too vague*. 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",... without ever telling us what *actually* is the "secret" or "truth" in the context of that specific story. Some real examples of this behavior are highlighted in red inside the Appendix A.1.1 and A.2.1, where it becomes clear that some sentences produced by the model are too ambiguous, too general, without describing what truly has happened in the narrative. On the other hand, GPT-4o, despite not being perfect, is far more specific and detailed when writing a narrative – see examples inside the Appendix A.1.3 and A.2.2.

In addition to the vagueness issue described above, GPT-3.5 writes, in general, lower quality stories with respect to GPT-4o. One of the limitations showed up when trying to combine movies: sometimes, instead of generating a new plot inspired by both parents, GPT-3.5 wrote two parallel narratives inside the same plot. One clear example of that phenomenon was observed when trying to combine the movies of *Mamma Mia!* and *Toy Story* – check this example in the Appendix A.1.2, where the narrative of *Toy Story* was colored in orange, while the part related to *Mamma Mia!* was depicted in blue (note that the two narratives never cross!).

To mitigate these undesired effects, other than changing the LLM to a better one (which could be also a valid solution), we can also apply prompting techniques to improve the response's quality. In fact, prompt engineering techniques had a very satisfactory effect when tuning the LLM to assign fitness scores, as no big difference was observed between GPT-3.5 and GPT-4o evaluations. With both language models, the prompting techniques described in 3.5 for the Inspiration f_{insp} and Instructions f_{inst} fitness functions worked as desired – check the examples in the Appendix A.3.3 to A.3.6. 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.

4.3 Analysis of the Evolution

This section focuses on comparing and discussing the evolution via SOO and MOO, its differences, and understanding whether our Genetic Algorithm (GA) is efficiently refining solutions over the generations. To do so, we needed to run some experiments in an attempt to draw conclusions. More specifically, we run our model using the same setups used for the survey (see Table 3.1).

In every case, we used the following combination of parameters – Number of generations (N_{gen}); Maximum no. of generations without improvement (M_{gen}); Population size (N); Probability of recombination (p_r); Probability of mutation (p_m); Elitism size (M); chosen language model (LLM) and Number of generated movie plots based on the user’s instructions (S):

N_{gen}	M_{gen}	N	p_r	p_m	M	LLM	S
15	3	15	0.7	0.3	3	GPT-4o	3

Table 4.1: Table with the setting of parameters for the experiments done to analyze evolution.

Six experiments were carried on, with 3 input movies each corresponding to the setup described in 3.1, using the parameters of Tab. 4.1. The following Table 4.2 encapsulates some of the results from our study. Note that due to computational costs, we had to limit both the population size to a relatively low number – just take a look at the “Time” column in the Table below. Later on, we will discuss the implications and potential advantages of increasing both the population size and the number of generations. The chosen language model was GPT-4o for both generation and evaluation, due to its better performance already discussed in the previous section 4.2.

Setup	Single-Objective Optimization			Multi-Objective Optimization		
	Top Fitness	Time (min)	Stopped at gen.	Top Fitness*	Time (min)	Stopped at gen.
1	7.78	56	15	7.30	40	10
2	7.60	42	11	7.09	33	9
3	7.58	47	11	7.59	34	9
4	7.74	47	11	7.50	39	10
5	7.32	41	12	6.80	26	8
6	7.74	29	7	7.53	36	8

Table 4.2: Results – best score, time and last generation number – considering six different initial setups.

*The best fitness score for the MOO case was calculated by taking a weighted sum of the fitnesses

by using the same values chosen for SOO – check 3.7.

By analyzing Table 4.2, we can draw some general conclusions regarding SOO and MOO optimizations. Firstly, we see that the top performances in both methods have quite similar fitness scores, with SOO performing generally slightly better. This is to be expected due to the way the fitness score is computed. In SOO, the 4-dimensional search space is reduced to a straight line, described by the equation $y = 2f_N + f_Q + 3f_{insp} + 3f_{inst}$ (recall equations 3.1 and 3.2). In contrast, MOO searches for an optimal solution inside a 4-dimensional space solution, attempting to maximize all the 4 dimensions simultaneously, rather than being confined to that specific straight line. Therefore, since the fitness score is based on the weights used in SOO's equation, it inherently favors higher fitness scores for SOO.

To fairly compare SOO and MOO, we need to analyze and compare the best individuals from both methods considering the whole space. Let us take as an example the two configurations from experiments 3 and 4. We will compare their best individuals projecting them into 2-dimensional spaces (4.1 and 4.2).

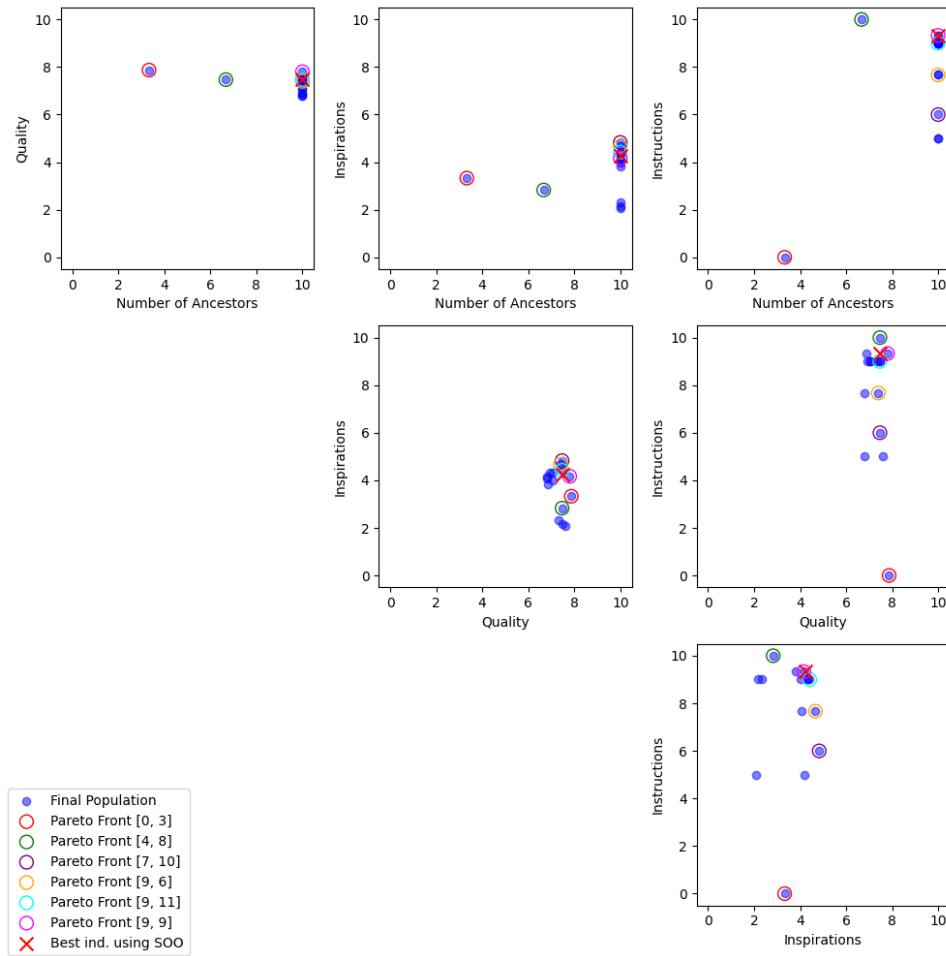


Figure 4.1: Setup No.3: Comparing SOO and MOO solutions in the objective space.

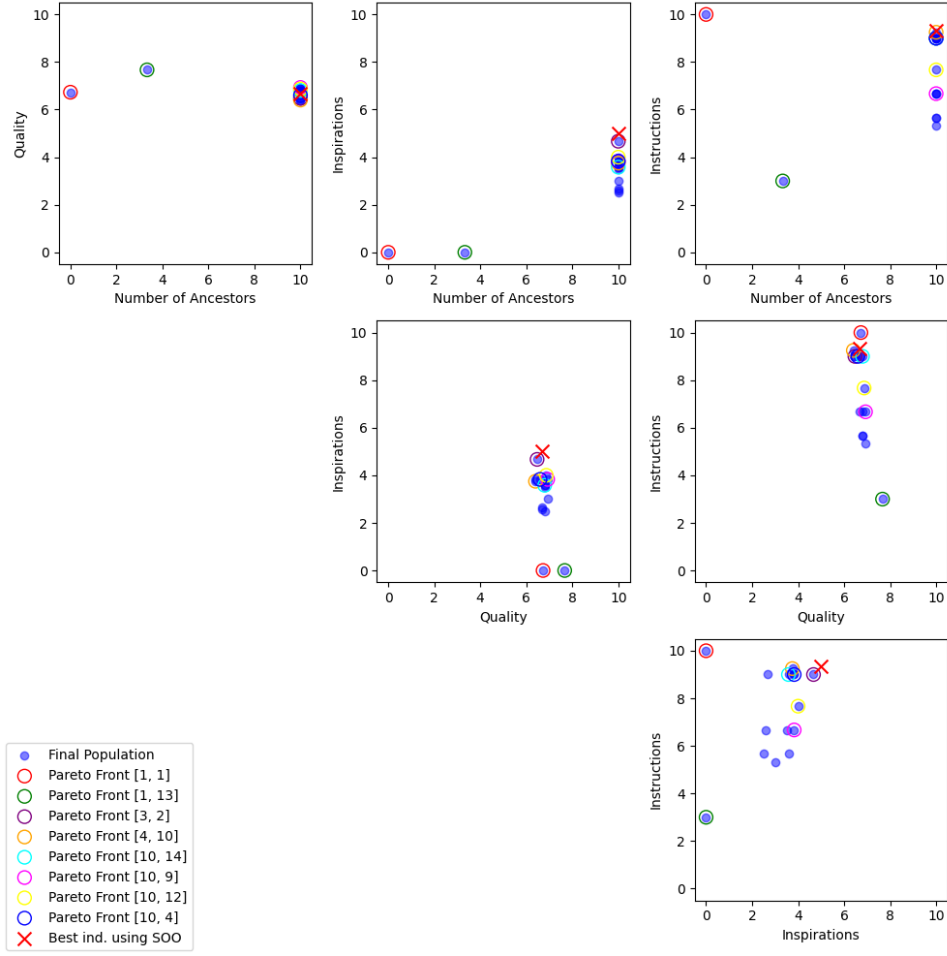


Figure 4.2: Setup No.4: Comparing SOO and MOO solutions in the objective space.

In Figures 4.1 and 4.2 the blue dots represent the final population generated with MOO, some of which have a colored circle around, meaning that they are part of the first Pareto front. On the caption, the first number a inside the identification $[a, b]$ represents the generation in which each individual was created, 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, representing the best individual achieved with SOO, 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. In some cases, in particular in Fig. 4.2, the red X dominates all Pareto front solutions in some of the 2-dimensional projections.

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 the user's instructions itself. MOO, on the other hand, will favor also extreme cases. 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 4.2, 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 **single**-objective optimization 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.

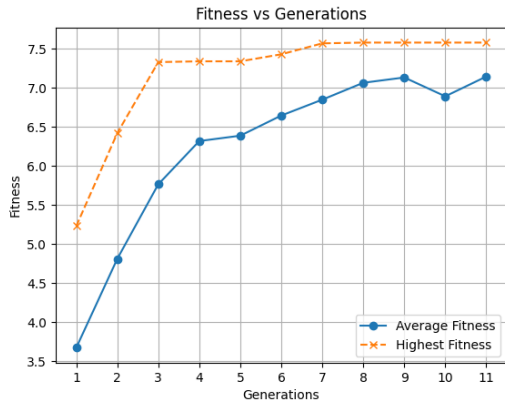
Still regarding Figures 4.1 and 4.2, we may notice a few more things:

- The objective "Number of Ancestors" (f_N) only gets discrete scores. This is because f_N counts what fraction of the chosen movies are ancestors of the generated movie (and scaled into the range from 0 to 10). Considering that we initialized these experiments with 3 existing movies, then the possible values that f_N may have are $0/3 \times 10 = 0$; $1/3 \times 10 = 3.33$; $2/3 \times 10 = 6.67$ and $3/3 \times 10 = 10$.
- The "Quality" objective f_Q always gets scores around 7, with almost always between 6 and 8. This LLM phenomenon was already discussed in Section 4.1, but may now be confirmed empirically.
- When comparing the four objective functions, the "Inspirations" is the one that gets, on average, poorer scores – in both examples no individual scored more than 6 points. This is not exclusive to these two 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.

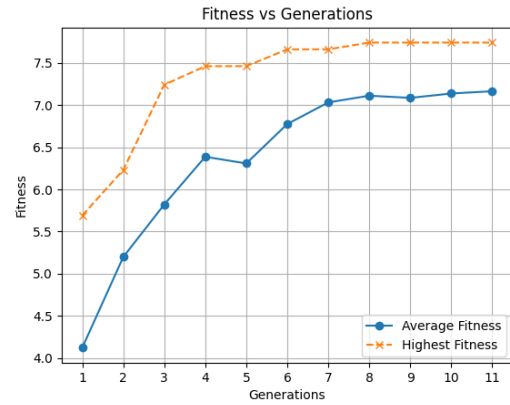
Finally, Figures 4.3(a) and 4.3(b) allow us to analyze how the population's fitness evolved through the generations. Note that we are considering the fitness relative to the SOO, obtained by calculating a weighted sum of the different objective functions.

In both cases, we observe a rapid growth of the fitness function in the first few generations, followed by a slow increase in the subsequent generations, which is a typical behavior of the evolution with GAs, indicating that convergence is being achieved. In both cases, the average and the highest fitness grow roughly at the same pace – the blue and orange curves have, approximately, the same shape.

Moreover, by analyzing the plots 4.3(a) and 4.3(b) we see that our stop criterion seems reasonable, as it appears that the algorithm has already reached a plateau (note that the GA stopped after 11 generations). In other words, our intuition is that running more generations would not produce significant improvements (this hypothesis will be further discussed in the next section 4.4). Regarding the stopping



(a) Setup No.3



(b) Setup No.4

Figure 4.3: Evolution of the fitness through generations with Single-Objective Optimization.

generation, by looking again at 4.2, we may conclude that MOO generally stops some generations before SOO. This may be due to the stricter stopping criterion in the SOO case (3 generations without improvement), compared to MOO, where slight improvements (smaller than a given δ) are still allowed over 3 generations. Nonetheless, if we had stopped SOO at the same generation as MOO, the final fitness score would remain unchanged most times, as it would stop less than 3 generations earlier, meaning that it had already achieved the plateau.

4.4 Increasing Complexity

In this Section, we tackle the questions that come naturally after the small experiments we performed: Is it worth running the algorithm with a larger **number of generations**? What if we increase the **population size**? What are the **computational implications** of both these actions?

We started by choosing one of the configurations used before – specifically, we chose setup no. 3 of Tables 3.1 and 4.2 – and we ran it for 45 generations ($N_{gen} = 45$), without any stopping criteria using both single and multi-optimization. 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 ($N_{gen} = 15$). In both cases, the parameter was increased by a factor of 3. It's important to note that we only changed one parameter at a time so that we can analyze its effects independently of the other variables. The algorithm was run in a machine that mounts a CPU Intel Xeon Processor E5-2609 v4, with 64GB of RAM. Results are summarized in Table 4.3.

*The best fitness score for the MOO case was calculated by taking a weighted sum of the fitnesses by using the same values chosen for SOO – check 3.7.

Setup	Single-Objective Optimization		Multi-Objective Optimization	
	Top Fitness	Time (min)	Top Fitness*	Time (min)
$N_{gen} = 45$	7.79	193	7.65	189
$N = 45$	7.9	242	7.79	222

Table 4.3: Results – best score and time – for an $N = 45$ (population size) and $N_{gen} = 45$

Once again, we confirm the tendency of SOO to perform better than MOO when we consider the top fitness score. However, as we had mentioned in 4.3, the computation of a single fitness will inherently favor SOO. Looking at Figures 4.4 and 4.5 is the most fair comparison between SOO and MOO solutions: we observe that the best individual generated with SOO (red X) generally dominates *most* of the MOO solutions, consistent with the conclusions drawn in the previous Section 4.3.

Note that in Figures 4.4 and 4.5 the blue dots represent the final population generated with MOO, some of which have a colored circle around, meaning that they are part of the first Pareto front. On the caption, the first number a inside the identification $[a, b]$ represent 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.

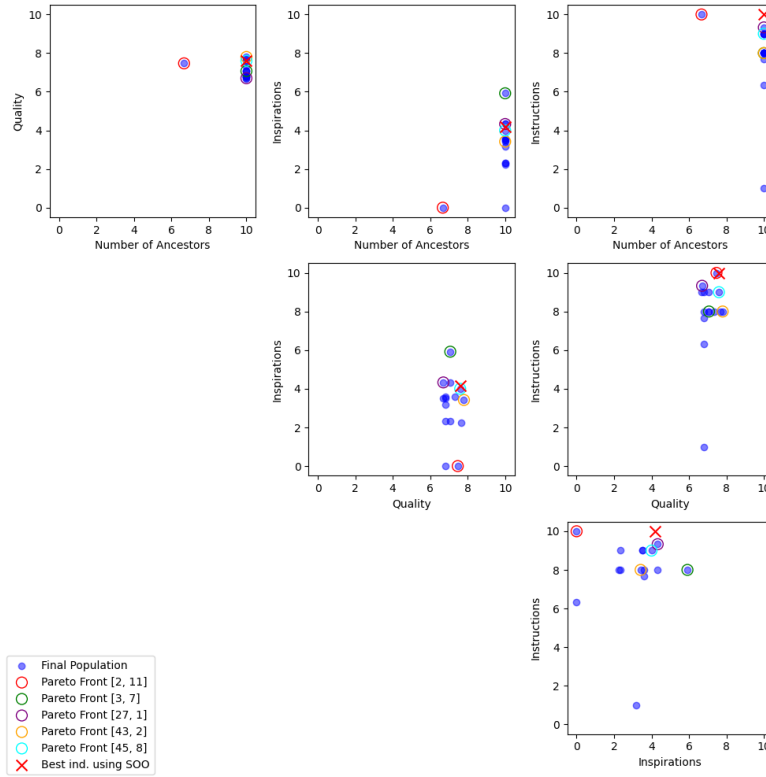


Figure 4.4: Comparing SOO and MOO solutions in the objective space with $N_{gen} = 45$

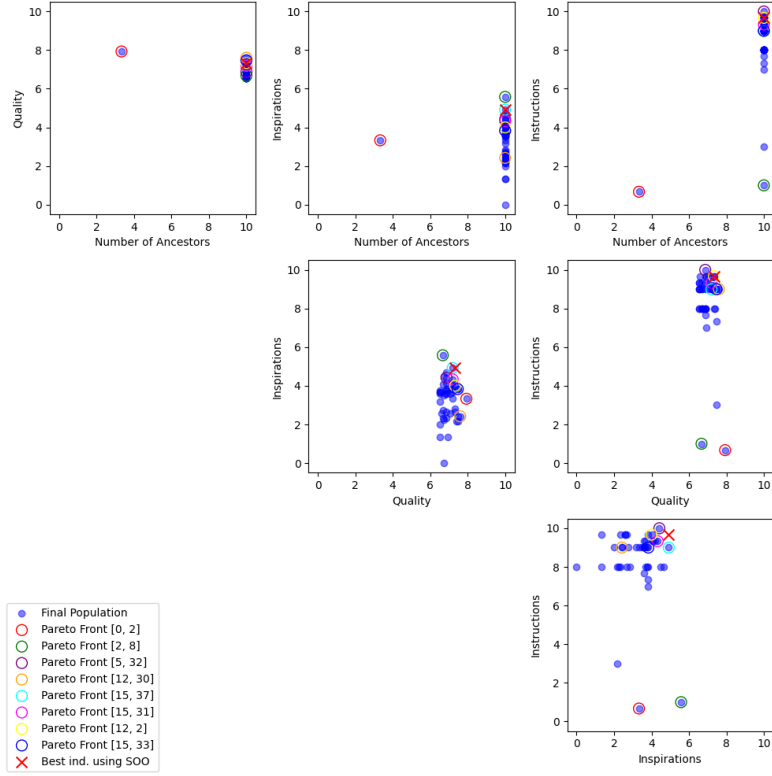
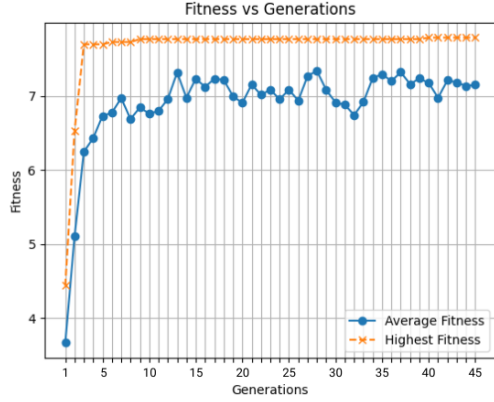


Figure 4.5: Comparing SOO and MOO solutions in the objective space with pop. size $N = 45$

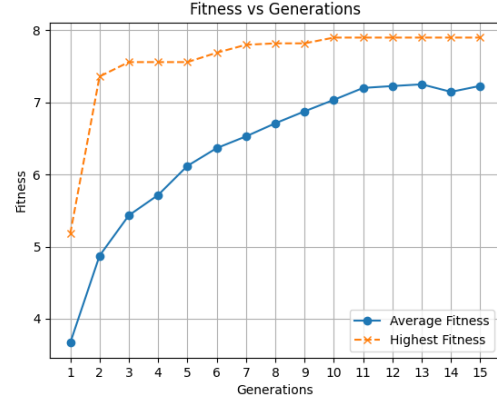
Analyzing the **scenario with $N_{gen} = 45$** , we see **no substantial improvement** in the algorithm's performance. When compared to the results in 4.2 (setup 3), the increase in the highest fitness in the SOO case was only 0.21. This is negligible considering a scale from 0 to 10, representing a growth of only 2.77% in the top fitness, despite the increase in the number of generations. 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 first scenario. This plateau is *very* clear in figure 4.6(a), which shows the evolution of the fitness function through the generations for the SOO case.

Our intuition is that iterating over too many generations will drive away new individuals from the information contained in the 'inspirational' movies, whereas stopping in the first generations will prevent the information contained in the original movies and in the instructions to spread and reach a large enough number of new movies. The ideal number of generations, therefore, is an intermediate range that balances the risk of losing connection with the 'inspirational' content while ensuring that the original ideas effectively propagate and evolve across new 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. Interestingly, when we look to the evolution of the fitness function



(a) $N_{gen} = 45$



(b) Pop. Size $N = 45$

Figure 4.6: Evolution of the fitness through generations with **Single-Objective Optimization** with N_{gen} and $N = 45$

over the generations (Fig. 4.6(b)), we see that the average fitness appears to constantly increase, even though the top fitness seems to already have reached stability.

Computational costs

Increasing N and N_{gen} have significant computational costs, as we can see by comparing the time spent in Table 4.3 (~ 200 minutes) with the previous experiments in Table 4.2 (~ 40 minutes). In fact, computational **complexity increases linearly** with both variables – let's understand how:

$\uparrow N_{gen}$: Performing 1 more generation in the GA implies computing $N - M$ more search operations (recombination or mutation), in order to generate enough individuals to populate the new generation (M is the elitism size). Once the whole population is created, each new individual undergoes evaluation through three operations that require the LLM corresponding to "quality", "inspirations" and "instructions" evaluations. Therefore, adding one generation involves performing $(N - M)$ search operations plus an additional $3(N - M)$ evaluation operations, resulting in $4(N - M)$ extra operations. Since this number does not depend on N_{gen} , we can say that the computational complexity increases linearly with N_{gen} , meaning the complexity is $O(N_{gen})$.

$\uparrow N$: similarly, adding 1 individual to the population size implies creating and evaluating one more individual every generation. This involves performing 1 more search operation (recombination or mutation) and 3 more evaluation operations, adding up to 4 more operations that require the LLM per generation or $4 \times N_{gen}$ operations in total. Therefore, the complexity is $O(N)$, meaning that the computational cost increases proportionally with N , with the proportional factor being $4 \times N_{gen}$.

Note that the previous analysis only accounted for the operations that required an LLM, as the rest of the code incurs a negligible computational cost. For instance, in the experiment with $N_{gen} = 45$ and

MOO (Table 4.3), out of the 189 minutes spent, 188.23 minutes were used in the LLM, occupying **more than 99%** of the execution time!

Considering the aforementioned example ($N_{gen} = 45$ and MOO – Table 4.3), we computed the average time for each different LLM operation (Table 4.4). Since the average times for all other experiments were very similar, this can be considered a representative example.

	Mutation	Recombination	Eval. Quality	Eval. Inspiration	Eval Instructions
Time (seconds)	10.8572	8.9049	4.5783	9.2034	2.4907

Table 4.4: Average time for each operation

Before proceeding, some clarification regarding Table 4.4 is necessary. Specifically, the time listed for evaluating 'inspirations' (~ 9 seconds) is greater because the evaluation loops through all the initially chosen movies. Since there were three 'inspiration' movies in this experiment, each evaluation operation took roughly 3 seconds.

We saw that, by doing these operations hundreds of times, the total time of a single experiment may become prohibitive. Our goal now is to understand which benefits would we gain if we had run the operations in parallel (for example, using GPU parallel computing).

Beginning with the search phase, in the worst-case scenario (i.e., the one that takes longer time), each individual will be created after a recombination and a mutation. Considering the times in Table 4.4, the creation of one individual would take $10.86 + 8.90 = 19.76$ seconds. Then, the evaluation phase consists of several different evaluation operations, with the 'Quality' evaluation being the longest (4.58 seconds). As mentioned before, evaluating 'Inspirations' can be divided in $S = 3$ different operations, each taking ~ 3 seconds. If all the evaluation operations are made in parallel, the total evaluation process for a single individual will take around 4.58 seconds.

In total, generating and evaluating one individual takes, in the worst case, $19.76 + 4.58 = 24.34$ seconds. If the idea is to use parallel computing, we can generate all individuals of a generation simultaneously, followed by parallel evaluation of all new individuals. Although each generation produces $N - M$ new individuals, parallel operations would reduce the time per generation to just 24.34 seconds. The spent time would be approximately:

$$24.34 \text{ seconds/generation} \quad (4.1)$$

Instead, without parallel operations, the total time is roughly given by:

$$(N - M) \times (19.76 + 4.58 + 9.20 + 2.49) = (N - M) \times 36.03 \text{ seconds/generation} \quad (4.2)$$

From equation 4.2 we can conclude that we would save the $9.20 + 2.49$ seconds by performing each

individual's evaluations in parallel. Additionally, we would save the multiplicative factor $(N - M)$ by parallelizing all individuals both during generation and evaluation. Using the example mentioned in this section ($N_{gen} = 45$ and MOO – Table 4.3, which took $\sim 189 \text{ min}$), which had $N = 15$ and $M = 4$, if we applied parallel computation as described, the experiment would take:

$$45 \times 24.34 \text{ seconds} = 1095,3 \text{ seconds} \approx 18 \text{ min} \quad (4.3)$$

More, considering the case with fewer generations ($N_{gen} = 15$), but with a population size of 45, the time savings gained by computing operations in parallel is even greater:

$$15 \times 24.34 \text{ seconds} = 365,1 \text{ seconds} \approx 6 \text{ min} \quad (4.4)$$

These substantial time reductions assume we have access to unlimited computational power to perform all the parallel operations. In conclusion, increasing complexity produces small improvements in the model's performance, while incurring significant time costs. However, parallelizing computations leads to notable time savings, reducing the time per generation to only 24.34 seconds. For instance, if we take the fastest experiment in Table 4.3 (189 minutes), it would take ~ 18 minutes if parallelized, representing a 90% time reduction!

4.5 Fitness Function Assessment with Human Feedback

This section discusses the results obtained with the surveys described in Section 3.8 designed to assess the quality of the fitness operation performed by the LLM. 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 built six different surveys, each one containing 4 artificially generated movies, and each survey is associated with 3 inspirational movies and one instruction to produce the new individuals (check Table 3.1). We obtained a total of 51 answers to the surveys distributed according to the following Table 4.5.

Survey No.	1	2	3	4	5	6	Total
No. Answers	6	10	9	7	10	9	51

Table 4.5: Number of answers per survey

Each set of movies was ranked 5 times: three considering their similarity with the 'inspiration' movies, one regarding the instructions, and the last considering the overall quality. The 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 and not any other arbitrary value. Computing a consensus ranking allowed us to directly compare the aggregated human evaluation with the evaluation generated by the LLM.

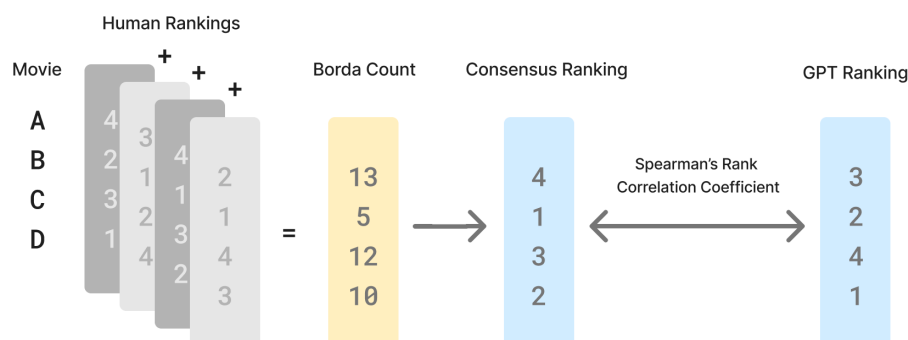


Figure 4.7: Computing rank correlation between Human and GPT evaluations

Figure 4.7 illustrates the process in which human feedback was collected to compare with the evaluation given by LLM. The ranking comparison involved using the **Spearman's ρ rank correlation** coefficient described in Chap. 3.8. This coefficient measures the correlation between two rankings, where 1 indicates perfect correlation, -1 means negative correlation and 0 means that there is no correlation between the two rankings. In addition to computing the rank correlation between human evaluations and those generated by GPT-3.5 and GPT-4o, we also decided to investigate the level of agreement among the individual human evaluators.

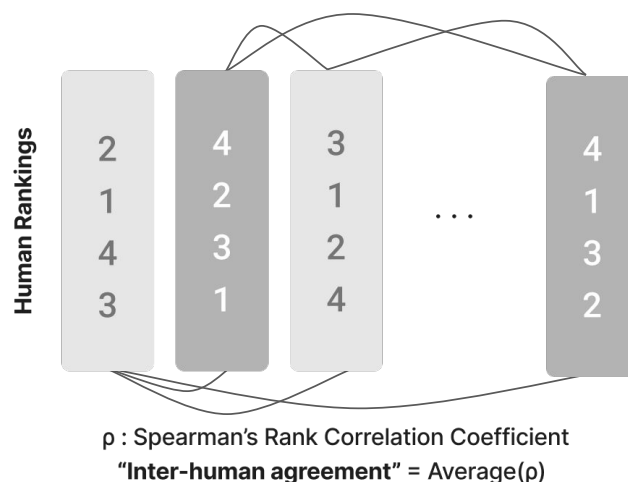


Figure 4.8: Computing Inter-human Agreement

To do so, we computed Spearman's ρ between every combination of pairs of human rankings and, then, averaged them. The idea is that the resulting number captures the **inter-human agreement**, in an interval from -1 to 1 . Figure 4.8 illustrates the calculation of the average ρ , which encapsulates the inter-human agreement.

The results of the human agreement with both GPT-3.5 and GPT-4o, along with the inter-human agreement, are presented in Tables 4.6 – 4.11. Each table corresponds to a single survey, with each row representing a different question from the survey, which in turn reflects a different human ranking. It is worth noting that, when our model is running, the LLM doesn't produce rankings, it assigns scores from 0 to 10. Once the scores are attributed to each movie, 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.

	Human agreement w/		Inter-human
	GPT-3.5	GPT-4o	agreement
<i>Cast Away</i>	0.40	0.00	0.49
<i>Godfather</i>	−0.40	−0.95	0.51
<i>Titanic</i>	−0.80	−0.63	0.37
Instructions	1.00	0.40	0.37
Quality	−0.40	0.40	−0.11

Table 4.6: Survey No.1

	Human agreement w/		Inter-human
	GPT-3.5	GPT-4o	agreement
<i>Inception</i>	−0.80	0.89	0.37
<i>Pinocchio</i>	0.95	0.80	0.76
<i>Matrix</i>	−0.20	0.00	0.20
Instructions	0.80	0.74	0.37
Quality	−0.32	0.40	0.54

Table 4.7: Survey No.2

	Human agreement w/		Inter-human
	GPT-3.5	GPT-4o	agreement
<i>Back to F.</i>	−0.77	−0.77	0.55
<i>Mamma M.</i>	0.63	0.40	0.48
<i>Toy Story</i>	−0.32	−0.11	0.60
Instructions	0.00	0.80	0.23
Quality	0.63	−0.80	0.09

Table 4.8: Survey No.3

	Human agreement w/		Inter-human
	GPT-3.5	GPT-4o	agreement
<i>Aladdin</i>	−0.95	−1.00	0.90
<i>Batman</i>	0.40	0.32	0.07
<i>Incredibles</i>	−0.20	−0.40	0.69
Instructions	0.80	0.95	0.57
Quality	−0.32	−0.20	0.30

Table 4.9: Survey No.4

	Human agreement w/		Inter-human
	GPT-3.5	GPT-4o	agreement
<i>Godzilla</i>	0.80	0.60	0.18
<i>Jurassic P.</i>	−0.80	−0.63	0.16
<i>King Kong</i>	0.11	0.32	0.44
Instructions	0.20	0.60	0.10
Quality	−0.26	−0.26	−0.08

Table 4.10: Survey No.5

	Human agreement w/		Inter-human
	GPT-3.5	GPT-4o	agreement
<i>Indiana J.</i>	0.74	0.74	0.10
<i>Pirates C.</i>	0.95	0.63	0.39
<i>Star Wars</i>	0.95	0.63	0.40
Instructions	0.40	0.00	0.32
Quality	−0.38	−0.38	−0.09

Table 4.11: Survey No.6

From Tables 4.6 – 4.11, we can draw some insights. Positive values, indicating a positive correlation, were highlighted in bold. Notably, we observe that the third column (inter-human agreement) contains

predominately high positive values, reflecting **strong correlation between different human evaluators**. Note that this is important to validate our study, as a 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 whether 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** (i.e., the first three rows of each table), the correlation factor does not show a clear pattern across the shown examples. Both GPT-3.5 and GPT-4o show varying degrees of correlation – ranging from strong, to zero, to negative correlations – with human rankings, depending on the specific 'inspiration' movie. Because this task is very dependent on the movies themselves, it is difficult to draw any general conclusion that fits most of the situations.
2. Generally, human and GPT evaluations align when determining which movies best follow the given **instructions**. This is evident by looking at the positive values in the fourth 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 rankings disagree often. One likely explanation is the limited validity of the GPT quality evaluation, which often assigns scores close to 7 or 8 (see 4.1), 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.

In order to get an overall picture of the results obtained, we built Table 4.12 that encapsulates the average results for each one of the objective functions. In particular, this table merges all 'inspiration' movies into one row, by computing their average value.

	Human agreement w/		Inter-human agreement
	GPT-3.5	GPT-4o	
Inspirations	0.04	0.05	0.42
Instructions	0.53	0.58	0.33
Quality	-0.17	-0.14	0.11

Table 4.12: Average values for the correlation between GPT and human rankings

Observing Table 4.12, we reiterate the conclusions drawn previously. Summing up, our findings regarding human feedback include:

- Humans evaluators generally agree with each other (Spearman's coefficient > 0);
- We found no correlation between human and GPT rankings for the 'Inspirations' objective function (Spearman's coefficient ~ 0);
- Human and GPT evaluations strongly align when considering the 'Instructions' objective function (Spearman's coefficient $\gg 0$);
- Human rankings generally disagree with GPT evaluations when considering the 'Quality' evaluation (Spearman's coefficient < 0);
- GPT-3.5 and GPT-4o show similar behavior (their correlation coefficients with human rankings do not show large differences).

Our aim, with this study, was to understand if the fitness scores assigned by the LLM aligned with human judgment. Our findings indicate that there is a **partial agreement** between the GPT's 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 which 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.

4.6 Conclusions

In conclusion, this chapter served the purpose of analyzing and discussing in detail the results produced by our tool, which efficiently evolved ideas through generations in an evolutionary context, despite the inherent limitations that naturally appear in an exploratory work like ours. We observed, as expected, that the GPT-4o model outperforms GPT-3.5. During evaluation, both models perform quite well when evaluating 'instructions', but perform poorer when it comes to deciding the overall quality of the new plots, a finding corroborated during our study with human feedback. In addition, we found out that the SOOs solution generally dominated most MOOs candidate solutions. Increasing complexity, such as the number of generations or population size, can be made efficiently by performing parallelized operations whenever possible. The next chapter (Chapter 5) builds upon the conclusions drawn in this chapter and presents some relevant reflections for future work.

5

Conclusions and Future Work

Contents

5.1	Conclusions	72
5.2	Future Work	74

In this Chapter, we will begin by summarizing all the key insights from our work in the Conclusions section 5.1, and conclude with some reflections and ideas for Future Work in section 5.2.

5.1 Conclusions

This project explored the possibility of integrating an LLM for the recombination, mutation, and evaluation steps of a Genetic Algorithm, where the individuals consist of pieces of text containing possible movie plots.

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 listed in this section.

We start by claiming that the architecture design described in 3.1 **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 4.4), as it yields significant time savings.

In brief, the **quality of the artificially generated movies** is highly dependent on the chosen LLM, but it is not the only relevant parameter. Below, we listed all the aspects that may influence the quality of the output, alongside a short conclusion on how that specific parameter affects the results. The studied parameters were:

- The **language model** (GPT-4o outperforms GPT-3.5);
- The **initial setup**, this is, the initial user guidelines, together with the provided list of existing movies that will serve as inspiration for our model (those parameters are decided by users and are dependent on their creative philosophy);
- The **optimization modality**: single- or multi-objective optimization (we found out that the SOO solution often dominates the majority of MOO Pareto solutions, but not necessarily all of them);
- The **population size** N (a very slight improvement of 4.22% on the top fitness was observed, when N was increased by a factor of 3);

- The **number of generations** N_{gen} (increasing significantly the number of generations produced a negligible improvement in the fitness of the best individual).

Finally, we conducted a study to assess the validity of the fitness scores that were being provided by the LLM, by asking people to rank a set of artificially generated movies in order to compare it with the LLM evaluation. With this study, we concluded that:

- Humans evaluators generally **agree** with each other.
- GPT-3.5 and GPT-4o **perform similarly** when evaluating.
- Human and GPT rankings generally agree when determining which movies better follow the given **instructions**; have a negative correlation when determining the **overall quality** of the movies; and don't correlate when determining how much the generated plots were **inspired** by the set of original films, varying for each scenario.

Limitations

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. It is evident, however, that those undesired phenomena are highly dependent on the chosen LLM. **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. More, logical inconsistencies are a frequent phenomenon, a type of LLM hallucination, and are difficult to solve. We found that **GPT-3.5** is much more susceptible of producing such undesired effects, when compared to more recent and robust models, like **GPT-4o**.

In the context of this work, the fact that the fitness function only agrees partially with the human feedback (assuming that the feedback is valid), suggests that the GA is not guiding the evolution of the individuals in the most optimal way. It is important to note, however, that the negative human-GPT rank correlations appear in the context of determining the general quality of a movie. Besides being a very subjective task (and the topic in which humans agreed the least), the LLM itself gave poorer answers in this context, which contributed to the disagreements observed.

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 are expected and inherent to any novel and exploratory approach like ours, leaving still room for Future Work – 5.2.

5.2 Future Work

In this work, we focused our work on generating new ideas specifically for movies. 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**, replacing the need for terminal interaction (Figure 3.3). Afterwards, **updating our film database** with more recent movie summaries would further improve the tool's usability, by addressing the current limitation of including only movies up to 2012. In addition, processing operations in parallel, as described in Section 4.4, would drastically reduce the runtime, further improving user satisfaction.

On a more technical note, future improvements could focus on refining the surveys to collect human feedback for further validation of the GA fitness function. Section 4.5 explored the possibility of the surveys being too exhaustive, with long and similar movie descriptions, making it quite complex to efficiently rank the different movies. The next logical step is to **simplify the surveys**, providing clearer and shorter movies to improve the quantity and quality of the responses.

Finally, as part of future work, consulting a **scriptwriting expert** would provide valuable insights to help design a better and task-specific tool – in particular, this would apply during the conceptualization phase when designing the structure of the movie plot or crafting recombination/mutation prompts.

Bibliography

- [1] D. Bamman, B. O'Connor, and N. A. Smith, "Learning latent personas of film characters," in *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, H. Schuetze, P. Fung, and M. Poesio, Eds. Sofia, Bulgaria: Association for Computational Linguistics, Aug. 2013, pp. 352–361. [Online]. Available: <https://aclanthology.org/P13-1035>
- [2] M. Awad and R. Khanna, *Machine Learning*. Berkeley, CA: Apress, 2015, pp. 1–18. [Online]. Available: https://doi.org/10.1007/978-1-4302-5990-9_1
- [3] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction, Second Edition (Springer Series in Statistics)*, 02 2009.
- [4] R. Sutton and A. Barto, *Reinforcement Learning: An Introduction*, ser. A Bradford book. MIT Press, 1998. [Online]. Available: <https://books.google.pt/books?id=CAFR6IBF4xYC>
- [5] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *The Bulletin of Mathematical Biophysics*, vol. 5, pp. 115–133, 12 1943.
- [6] Y. Wang, Y. Li, Y. Song, and X. Rong, "The influence of the activation function in a convolution neural network model of facial expression recognition," *Applied Sciences*, vol. 10, no. 5, 2020. [Online]. Available: <https://www.mdpi.com/2076-3417/10/5/1897>
- [7] M. Minsky and S. Papert, *Perceptrons*. Cambridge, MA: MIT Press, 1969.
- [8] K. Hornik, M. B. Stinchcombe, and H. L. White, "Multilayer feedforward networks are universal approximators," *Neural Networks*, vol. 2, pp. 359–366, 1989. [Online]. Available: <https://api.semanticscholar.org/CorpusID:2757547>
- [9] G. F. Montufar, R. Pascanu, K. Cho, and Y. Bengio, "On the number of linear regions of deep neural networks," in *Advances in Neural Information Processing Systems*, Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Weinberger, Eds., vol. 27. Curran

- Associates, Inc., 2014. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2014/file/109d2dd3608f669ca17920c511c2a41e-Paper.pdf
- [10] C. M. Bishop and H. Bishop, *Deep Learning*. Springer International Publishing, 2024.
- [11] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin, “Attention is all you need,” in *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., vol. 30. Curran Associates, Inc., 2017. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf
- [12] G. Sanderson, “But what is a gpt? visual intro to transformers — chapter 5, deep learning,” YouTube, April 2024. [Online]. Available: <https://www.youtube.com/watch?v=wjZofJX0v4M&t=1383s>
- [13] “Tokenizer - learn about language model tokenization,” 6 2024. [Online]. Available: <https://platform.openai.com/tokenizer>
- [14] C. Li, “Openai’s gpt-3 language model: A technical overview,” 6 2020. [Online]. Available: <https://lambdalabs.com/blog/demystifying-gpt-3>
- [15] T. Mikolov, W.-t. Yih, and G. Zweig, “Linguistic regularities in continuous space word representations,” in *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, L. Vanderwende, H. Daumé III, and K. Kirchhoff, Eds. Atlanta, Georgia: Association for Computational Linguistics, Jun. 2013, pp. 746–751. [Online]. Available: <https://aclanthology.org/N13-1090>
- [16] J. Alammr, “The illustrated transformer,” 6 2018. [Online]. Available: <https://jalammar.github.io/illustrated-transformer/>
- [17] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei, “Language models are few-shot learners,” in *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, Eds., vol. 33. Curran Associates, Inc., 2020, pp. 1877–1901. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2020/file/1457c0d6bfc4967418bfb8ac142f64a-Paper.pdf
- [18] G. Sanderson, “Attention in transformers, visually explained — chapter 6, deep learning,” YouTube, April 2024. [Online]. Available: <https://www.youtube.com/watch?v=eMlx5fFNoYc&t=982s>

- [19] J. Kaplan, S. McCandlish, T. Henighan, T. B. Brown, B. Chess, R. Child, S. Gray, A. Radford, J. Wu, and D. Amodei, "Scaling laws for neural language models," *ArXiv*, vol. abs/2001.08361, 2020. [Online]. Available: <https://api.semanticscholar.org/CorpusID:210861095>
- [20] U. Eric, "Prompt engineering techniques," 2 2024. [Online]. Available: <https://learn.microsoft.com/en-us/azure/ai-services/openai/concepts/advanced-prompt-engineering>
- [21] E. Saravia, "Prompt Engineering Guide," 12 2022. [Online]. Available: <https://github.com/dair-ai/Prompt-Engineering-Guide>
- [22] L. Weng, "Prompt engineering," *lilianweng.github.io*, Mar 2023. [Online]. Available: <https://lilianweng.github.io/posts/2023-03-15-prompt-engineering/>
- [23] S. Sander, "Prompt engineering guide," 2022. [Online]. Available: <https://learnprompting.org/docs/intro>
- [24] J. Wei, X. Wang, D. Schuurmans, M. Bosma, E. H. Hsin Chi, F. Xia, Q. Le, and D. Zhou, "Chain of thought prompting elicits reasoning in large language models," *ArXiv*, vol. abs/2201.11903, 2022. [Online]. Available: <https://api.semanticscholar.org/CorpusID:246411621>
- [25] T. Kojima, S. S. Gu, M. Reid, Y. Matsuo, and Y. Iwasawa, "Large language models are zero-shot reasoners," *ArXiv*, vol. abs/2205.11916, 2022. [Online]. Available: <https://api.semanticscholar.org/CorpusID:249017743>
- [26] M. A. Iqbal, "Limitations of large language models for real-world applications," 1 2024. [Online]. Available: <https://medium.com/@muhammad.amir.iqbal/limitations-of-large-language-models-for-real-world-applications-14a6889cda04>
- [27] M. Reeves, A. Job, and M. Burtsev, "The working limitations of large language models," *Frontiers - MIT Sloan Management Review*, 11 2023. [Online]. Available: <https://sloanreview.mit.edu/article/the-working-limitations-of-large-language-models/>
- [28] Z. Xu, S. Jain, and M. Kankanhalli, "Hallucination is inevitable: An innate limitation of large language models," 2024. [Online]. Available: <https://arxiv.org/abs/2401.11817>
- [29] D. B. Fogel, D. Liu, and J. M. Keller, *Fundamentals of Computational Intelligence*. Wiley, 6 2016.
- [30] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: Nsga-ii," *IEEE Transactions on Evolutionary Computation*, vol. 6, pp. 182–197, 4 2002. [Online]. Available: <https://ieeexplore.ieee.org/document/996017>
- [31] J. Branke, K. Deb, K. Miettinen, and R. Słowiński, Eds., *Multiobjective Optimization*. Springer Berlin Heidelberg, 2008, vol. 5252.

- [32] J. Lehman, J. Gordon, S. Jain, K. Ndousse, C. Yeh, and K. O. Stanley, "Evolution through large models," 6 2022. [Online]. Available: <http://arxiv.org/abs/2206.08896>
- [33] E. Meyerson, M. J. Nelson, H. Bradley, A. Gaier, A. Moradi, A. K. Hoover, and J. Lehman, "Language model crossover: Variation through few-shot prompting," 2024. [Online]. Available: <https://arxiv.org/abs/2302.12170>
- [34] H. Bradley, A. Dai, H. Teufel, J. Zhang, K. Oostermeijer, M. Bellagente, J. Clune, K. Stanley, G. Schott, and J. Lehman, "Quality-diversity through ai feedback," 10 2023. [Online]. Available: <http://arxiv.org/abs/2310.13032>
- [35] H. Bradley, H. Fan, T. Galanos, R. Zhou, D. Scott, and J. Lehman, *The OpenELM Library: Leveraging Progress in Language Models for Novel Evolutionary Algorithms*. Singapore: Springer Nature Singapore, 2024, pp. 177–201. [Online]. Available: https://doi.org/10.1007/978-981-99-8413-8_10
- [36] A. Chen, D. M. Dohan, and D. R. So, "Evoprompting: Language models for code-level neural architecture search," 2023. [Online]. Available: <https://arxiv.org/abs/2302.14838>
- [37] C. Xu, Q. Sun, K. Zheng, X. Geng, P. Zhao, J. Feng, C. Tao, and D. Jiang, "Wizardlm: Empowering large language models to follow complex instructions," 2023. [Online]. Available: <https://arxiv.org/abs/2304.12244>
- [38] P. L. Lanzi and D. 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*, ser. GECCO '23. ACM, Jul. 2023. [Online]. Available: <http://dx.doi.org/10.1145/3583131.3590351>
- [39] A. Madaan, N. Tandon, P. Gupta, S. Hallinan, L. Gao, S. Wiegrefe, U. Alon, N. Dziri, S. Prabhume, Y. Yang, S. Gupta, B. P. Majumder, K. Hermann, S. Welleck, A. Yazdanbakhsh, and P. Clark, "Self-refine: Iterative refinement with self-feedback," 3 2023. [Online]. Available: <http://arxiv.org/abs/2303.17651>
- [40] F. Dayo, . Ahmed, A. Memon, and . N. Dharejo, "Scriptwriting in the age of ai: Revolutionizing storytelling with artificial intelligence," *Journal of Media Communication (JMC) (JMC)*, vol. 4, p. 2023. [Online]. Available: <https://jmc.ilmuainiversity.edu.pk/arc/Vol4/4.1/2.pdf>
- [41] "Shortlyai — your ai writing partner." [Online]. Available: <https://www.shortlyai.com/>
- [42] "Free scriptwriting software — nolanai." [Online]. Available: <https://www.nolanai.app/>
- [43] "Best ai scriptwriter in 2024 — squibler." [Online]. Available: <https://www.squibler.io/ai-script-writer>

- [44] "Create engaging scripts with ai script generator." [Online]. Available: <https://simplified.com/ai-script-generator>
- [45] S. Field, *Screenplay: The Foundations of Screenwriting*, ser. A Delta book. Delta Trade Paperbacks, 2005. [Online]. Available: https://books.google.pt/books?id=Rqx2S_fJ9YIC
- [46] I.-S. Kim and W. McLean, "Computing the hausdorff distance between two sets of parametric curves," *Communications of The Korean Mathematical Society*, vol. 28, pp. 833–850, 2013. [Online]. Available: <https://api.semanticscholar.org/CorpusID:55134429>
- [47] O. Schutze, X. Esquivel, A. Lara, and C. A. C. Coello, "Using the averaged hausdorff distance as a performance measure in evolutionary multiobjective optimization," *IEEE Transactions on Evolutionary Computation*, vol. 16, no. 4, pp. 504–522, 2012. [Online]. Available: <https://ieeexplore.ieee.org/document/6151115>
- [48] T. W. MacFarland and J. M. Yates, *Spearman's Rank-Difference Coefficient of Correlation*. Cham: Springer International Publishing, 2016, pp. 249–297. [Online]. Available: https://doi.org/10.1007/978-3-319-30634-6_8



Appendix A: Examples

The Appendix supports the analysis of chapter 4. It contains:

- A.1 – A.2: a set of examples of artificially generated movie plots that come from the LLM output of both **Recombination** and **Mutation** Operations for both models **GPT-3.5** and **GPT-4o**
- A.3: examples of the LLM output of the **Evaluation** process, one for each the three metrics f_Q , f_{insp} and f_{inst} , for both language models.

A.1 Recombination

A.1.1 Recombination performed by GPT-3.5: *Mamma Mia!* + *Back to the Future*

Text in red refer to the ambiguity and vagueness present in the LLM output, discussed in Section 4.2.

Title: "Timeless Love on Kalokairi Island"

Genre: Sci-Fi / Romance / Musical

World-Building: *Time and Place:* The story unfolds in the charming town of Kalokairi, Greece, blending the picturesque island setting with the futuristic elements of time travel. *Backstory:* Kalokairi is known for its legendary fountain of Aphrodite,

rumored to have mystical powers that influence love and destiny. The island has a rich history of love stories and secrets waiting to be uncovered.

Plot:

First Act: Marty, a young musician with dreams of rock stardom, stumbles upon a mysterious scientist, Dr. Emmett, who unveils a time-traveling DeLorean. Marty's journey back to 1955 coincides with Sophie's quest to discover her father's identity on Kalokairi. Marty's interference in his parents' past love story inadvertently affects Sophie's family dynamics, leading to a ripple effect across time.

First Turning Point: Marty's attempt to set up George and Lorraine mirrors Sophie's efforts to connect with her potential fathers, causing unexpected twists in both timelines.

Second Act: Marty and Sophie **navigate through challenges** in their respective quests, **facing dilemmas** of altering the past and uncertain futures. The bond between Marty and Sophie strengthens as they realize the importance of family, love, and the impact of their actions on those around them.

Second Turning Point: Marty's realization of the consequences of changing history parallels Sophie's emotional turmoil as she **grapples with the truth** about her father's identity.

Climax: Marty and Sophie must confront the **ultimate test of love and sacrifice** as they race against time to set things right in their intertwined destinies. The fusion of Marty's time-traveling adventure and Sophie's quest for family truth **culminates in a heartwarming and musically enchanting climax**.

Resolution: Marty and Sophie find closure and newfound understanding in their personal journeys, leading to a harmonious blend of past, present, and future on Kalokairi Island. The power of love, forgiveness, and self-discovery transforms their lives and those around them, creating a timeless bond that transcends generations.

Main Characters: *Protagonist:* Marty: A passionate young musician with a heart full of dreams, courage, and a desire to make things right in his family's history. He learns the value of love and connection through his time-traveling adventure.

Sophie: A determined and compassionate young woman on a quest to uncover her true identity and find her place in the world. Her journey of self-discovery intertwines with Marty's, leading to revelations about love and family bonds.

Other Main Characters:

- Dr. Emmett: The eccentric scientist who guides Marty on his time-traveling journey, offering wisdom and support in moments of uncertainty.

- Donna: The vibrant and spirited villa owner on Kalokairi, whose past secrets and lost loves resurface as Sophie's quest unfolds, revealing layers of her character and resilience.

- George and Lorraine: Characters inspired by Marty's parents, embodying the essence of young love, second chances, and the power of destiny in shaping their intertwined futures.

A.1.2 Recombination performed by GPT-3.5: *Mamma Mia!* + *Toy Story*

This example supports the analysis in Section 4.2, where the text produced reflects two parallel narratives (that never cross!), instead of one cohesive narrative. The two narratives refer to *Mamma Mia!* (blue) and *Toy Story* (in orange).

Title: "Toy Island Wedding Adventure"

Genre: Adventure / Family / Romance

World-Building: *Time and Place:* Present day on the enchanting Toy Island, a place where toys come to life when humans are not around, and a Greek island named Kalokairi known for its beauty and romantic atmosphere. *Backstory:* Toy Island is a magical place where toys have their own adventures and friendships, while Kalokairi holds the legendary fountain of Aphrodite, symbolizing love and new beginnings.

Plot:

First Act: On Toy Island, Woody, the brave cowboy doll, leads a group of toys preparing for Andy's family's move. They discover new toys arriving, including Buzz Lightyear, a space ranger action figure. Meanwhile, on Kalokairi, Sophie prepares for her wedding, inviting three potential fathers without her mother's knowledge, hoping to uncover her true parentage. Woody and Buzz face a misunderstanding that leads to Buzz getting lost, while Sophie navigates the arrival of her three fathers on the island.

First Turning Point: Woody and Buzz find themselves captured by Sid, the mischievous neighbor on Toy Island, while Sophie grapples with the revelation of her fathers' identities on Kalokairi.

Second Act: Woody and Buzz must escape Sid's clutches with the help of the island's friendly toys, facing challenges and learning the true value of friendship. Sophie confronts her feelings and fears about her upcoming wedding, seeking guidance from her mother and the unexpected bond forming with her fathers.

Second Turning Point: Woody and Buzz, along with the island toys, confront Sid and save the day, while Sophie makes a decision about her future and her relationships with her fathers.

Climax: A heartwarming celebration ensues on Toy Island and Kalokairi, as Woody, Buzz, Sophie, and her family come together to embrace love, friendship, and new beginnings.

Resolution: Woody and Buzz bid farewell to Toy Island, knowing they have each other's backs, while Sophie embarks on a new journey with her chosen family and a postponed wedding, filled with love and adventure.

Main Characters: *Protagonist:* - Woody: A loyal and courageous cowboy doll, always looking out for his toy friends and learning the importance of acceptance and teamwork.

- Sophie: A determined young woman on a quest for her true parentage, discovering the power of family and love on the enchanting island of Kalokairi.

Other Main Characters:

- Buzz: An adventurous space ranger toy who learns the value of being a part of a team and finding his place in the toy world.
- Donna: Sophie's caring mother, navigating past relationships and new beginnings on Kalokairi.
- Sid: The mischievous neighbor on Toy Island, representing challenges and obstacles that must be overcome through unity and courage.

A.1.3 Recombination performed by GPT-4o: *Mamma Mia!* + *Back to the Future*

Title: "Time on the Island"

Genre: Adventure / Musical / Family

World-Building: *Time and Place:* Present day on a beautiful island called Kalokairi, Greece, and in the town of Hill Valley, California. *Backstory:* Kalokairi is known for its stunning beaches and the legendary fountain of love, while Hill Valley is a small town with a history of family struggles and dreams of a brighter future.

Plot:

First Act: A sunny day on Kalokairi. Sophie, a 20-year-old bride-to-be, is excitedly preparing for her wedding but feels unsure about her father. Sophie finds her mother's diary and learns she has three possible fathers. She decides to invite them to her wedding without telling her mother, hoping to discover the truth. Meanwhile, in Hill Valley, a 17-year-old boy named Max dreams of being a rock musician. He feels stuck in his dull life and wishes for adventure. Max meets his quirky friend, Doc, who reveals he has built a time machine. They decide to travel to Kalokairi to help Sophie and find out about her father.

First Turning Point: Max and Doc arrive on Kalokairi just as Sophie's three potential fathers arrive. Max bumps into Sophie, and they share a moment of connection.

Second Act: Sophie hides the three men in her villa, while Max and Doc explore the island. They learn about the fountain of love and its magical powers. Sophie sails with her potential fathers on a boat, hearing stories about her mother's past. Max joins them, and they sing songs about love and adventure. Tension rises as Sophie struggles to choose her father. Meanwhile, Max

tries to impress Sophie with his music but feels nervous. At a party, Sophie's mother, Donna, performs with her friends, and Max gets inspired to play his guitar. He decides to sing a song to Sophie.

Second Turning Point: Just as Max is about to sing, the three fathers argue about who should give Sophie away. Sophie feels overwhelmed and runs away. Max follows Sophie and encourages her to be true to herself. They share their dreams and fears, and Sophie realizes she needs to be honest with her mother.

Climax: At the wedding, Sophie stands at the altar, unsure. Max steps in and plays a heartfelt song, bringing everyone together. Sophie confesses her secret to her mother and the three fathers. They all realize that love is what truly matters, not just blood ties.

Resolution: The wedding turns into a celebration of love, with everyone dancing and singing. Sophie decides to travel the world with Max instead of getting married right away. Max and Sophie sail away together, promising to chase their dreams and discover their true selves.

Main Characters: *Protagonist:* Sophie, a 20-year-old bride-to-be who is kind-hearted and adventurous. She wants to find her father and avoid her mother's mistakes. Her flaw is her fear of commitment.

Antagonist: Not applicable.

Other Main Characters:

- Max: A 17-year-old dreamer who wants to be a rock musician. He is brave, creative, and supportive. His goal is to help Sophie and find his own path.

- Doc: A quirky scientist who loves adventure and helps Max with the time machine. He is wise and funny, always ready for a new challenge.

- Donna: Sophie's mother, a strong woman who has a complicated past. She learns to embrace love and support her daughter's choices.

A.2 Mutation

A.2.1 Mutation performed by GPT-3.5: *Mamma Mia!*

Mutation: *"Subvert genre expectations by introducing elements that defy traditional genre conventions"*

Text in red refers to the ambiguity and vagueness present in the LLM output, discussed in Section 4.2.

Title: Secrets of the Aegean

Genre: Romantic Comedy, Musical

World-Building: *Time and Place:* The story takes place in the present day on the fictional island of Aegea, known for its crystal-clear waters, vibrant culture, and ancient legends. *Backstory:* Aegea has a rich history steeped in mythology, with tales of love, betrayal, and redemption echoing through the generations. The island's fabled fountain of Aphrodite is said to have mystical powers that can reveal true love and hidden truths.

Plot:

First Act: Mia, along with her loyal friend Elena, arrives on the enchanting island of Aegea for her wedding. She confides in Elena about her quest to uncover the identity of her father among three potential candidates.

First Turning Point: As Mia delves deeper into her investigation, she encounters Marcus, a suave businessman with a hidden agenda that threatens to derail her plans. She must navigate his manipulative schemes while also dealing with the unexpected arrival of her three possible fathers.

Second Act: Mia seeks the truth from each of the men in secret, leading to revelations about her mother's past and her own doubts about the upcoming wedding. Meanwhile, Marcus's actions become increasingly sinister, putting Mia's happiness at risk.

Second Turning Point: On the day of the wedding, Mia is faced with a choice between following her heart or succumbing to Marcus's deceptions. As **secrets are revealed** and **tensions reach a boiling point**, Mia must confront her fears and make a decision that will change her life forever.

Climax: With the help of Nikos and Elena, Mia confronts Marcus and **uncovers his true intentions**, leading to a final showdown that resolves the conflict once and for all.

Resolution: In a heartwarming conclusion, Mia discovers the identity of her father, finds unexpected love in the most unlikely place, and learns that true happiness comes from following one's heart. The island of Aegea witnesses a celebration of love, friendship, and new beginnings as Mia's journey comes to a joyous end.

Main Characters: *Protagonist:* Mia, a free-spirited young woman who is about to get married on the picturesque island of Aegea. Mia is adventurous, curious, and determined. She is searching for her true father among three potential candidates to avoid repeating her mother's mistakes. Mia's main goal is to uncover the truth about her parentage and find her own path to happiness.

Antagonist: Marcus, a charming but manipulative businessman who has a hidden agenda on the island. Marcus is a smooth talker with a dark side, using his power and influence to get what he wants. He sees Mia's wedding as an opportunity to further his own interests and will stop at nothing to achieve his goals.

Other Main Characters: - Elena: Mia's best friend and bridesmaid, who is fiercely loyal and always has Mia's back. She is a source of support and comic relief throughout the story. - Nikos: A local tavern owner who harbors a secret crush on Mia. He is kind-hearted, hardworking, and plays a key role in helping Mia navigate the challenges she faces.

A.2.2 Mutation performed by GPT-4o: *Mamma Mia!*

Mutation: *"Introduce unexpected plot twists or revelations that change the direction of the story"*

Title: Hidden Ties

Genre: Romantic Comedy

World-Building: *Time and Place:* A picturesque Mediterranean island named Aeloria during the present day, where the vibrant culture and warm climate shape the lives of its inhabitants. *Backstory:* Aeloria was once a thriving trading hub in ancient times, known for its luxurious goods and artistic heritage. However, a devastating storm centuries ago caused many to abandon the island, leaving behind only tales of its splendor. In recent years, the island has been revitalized by tourism, with a focus on its mythical past, including legends of a hidden treasure said to grant wishes to those who find it.

Plot:

- *First Act:* - 20-year-old bride-to-be Lila discovers a hidden diary belonging to her mother, revealing that she has three potential fathers. Curious and determined to uncover the truth, she invites them to her upcoming wedding without her mother's knowledge. - Lila's best friends, Mia and Clara, arrive on Aeloria, excited for the wedding but unaware of Lila's secret mission. - Lila's mother, Elena, is preparing for the wedding, blissfully unaware of the chaos about to unfold. She reminisces about her past with her two best friends, who help her manage the villa and its finances.

- *First Turning Point:* - The three potential fathers—charming architect Daniel, rugged explorer Erik, and suave businessman Oliver—arrive on the island. Lila manages to keep them hidden from Elena while navigating her feelings and the pressure of her impending nuptials. - Lila learns from Erik that Elena used to dream of traveling the world, sparking doubts about her own decision to marry young. She begins to question if she's making a mistake.

- *Second Act:* - As the wedding approaches, Lila spends time with each father figure, learning about their lives and her mother's past. Each man believes he is the one who should give her away, leading to tension. - During a night of celebration, Lila overhears a conversation revealing that her mother's villa is built on the ruins of an ancient temple, where a treasure is rumored to be hidden. This treasure could change everything for them financially. - Tensions rise as Elena discovers the men's presence

and confronts Lila. In a heated argument, Elena reveals that she never wanted to get married and had always hoped Lila would choose a different path.

- *Second Turning Point*: - Lila decides to confront her mother about the truth behind her father's identity instead of proceeding with the wedding. They have a heartfelt conversation, where Elena admits she has always been afraid of love and commitment. - In a surprising twist, Lila discovers that the treasure is not a physical object but rather a long-lost family heirloom that could provide answers about her father's identity.

- *Climax*: - On the day of the wedding, Lila decides to postpone it. As she prepares to reveal the truth to her mother and the three men, a storm hits Aeloria, causing chaos at the venue. - In the midst of the storm, the truth about each man's connection to Elena and Lila's family is revealed. Each man shares a memory that ties them to Lila's mother in a meaningful way, leading to a powerful moment of revelation.

- *Resolution*: - Lila, now understanding her mother's fears and desires, decides to embark on a journey to find the family heirloom. She proposes a trip with her friends and potential fathers to explore their roots rather than rushing into marriage. - Elena and Daniel rekindle their romance, while Erik and Oliver decide to support Lila in her quest, forming a new bond as a family. - The film ends with Lila, Elena, and the three men setting sail together, ready to uncover the secrets of their past and embrace their future.

Main Characters:

- *Protagonist*: Lila Torres - A 20-year-old with a passion for adventure and a heart full of dreams. Lila is curious, determined, and sometimes impulsive, which leads her to invite her potential fathers. She struggles with self-doubt and the weight of expectations but ultimately desires to forge her own path.

- *Antagonist*: Elena Torres - Lila's mother, who represents the fears of commitment and the weight of past mistakes. Elena is strong-willed and fiercely protective but has a complicated relationship with love due to her own unresolved issues. Her journey is about overcoming her fears and accepting that love can be different from what she once believed.

Other Main Characters:

- Daniel Hayes: A charming architect from the city, who struggles with his own fear of commitment but is drawn to Elena's past. He hopes to reconnect with her and build a future together;

- Erik Lindström: A rugged explorer with a free spirit, who represents the adventurous side of life. He encourages Lila to pursue her dreams and challenges her to break free from societal expectations;

- Oliver Grant: A suave businessman who initially appears to be the most stable choice but reveals a deeper, more vulnerable side. He is motivated by a desire to be part of a family and fears losing that connection;

- Mia and Clara: Lila's best friends, who provide comic relief and support throughout her journey. They each have their own arcs that involve discovering love and purpose on Aeloria.

A.3 Evaluation

A.3.1 Evaluating Quality f_Q with GPT-3.5

The evaluations consist in **score + brief explanation** for each criterion **A** to **O** (see 3.5).

Here, we present a generic example to illustrate the output of the LLM 'Quality' evaluation.

A. **9** The plot is well-structured and coherent, with a clear progression of events and connections between different story elements.

B. **8** While the concept of time travel and family curses is not entirely new, the combination of these elements and the specific details of the plot make it quite original.

- C. **7** The characters' traits are consistent throughout the story, with their actions and decisions aligning with their established personalities.
- D. **8** The plot evokes a good emotional response, especially through themes of family, connection, and overcoming challenges.
- E. **6** The plot structure is moderately complex, with time travel elements and multiple layers of mystery and discovery.
- F. **8** Character development is well-done, particularly with Lily's growth and the complexities of her relationships with her parents.
- G. **9** The worldbuilding is rich and immersive, with detailed descriptions of the town, the family history, and the magical elements.
- H. **8** The pacing of the plot keeps the story engaging and moves at a suitable speed to maintain interest.
- I. **7** The themes explored, such as family legacy, sacrifice, and the power of connection, are original and thought-provoking.
- J. **9** The narrative is cohesive, with all plot points and character arcs coming together in a satisfying way.
- K. **7** While some plot twists are predictable, there are enough unexpected developments to keep the audience engaged.
- L. **8** The plot offers potential for visually captivating scenes, especially during the time travel sequences and magical moments.
- M. **6** There is some effectiveness in embedding social and emotional commentary within the plot, though it could be further developed.
- N. **8** The cultural relevance of family secrets, generational curses, and the importance of history and heritage adds depth to the plot.
- O. **7** Symbolism and metaphor are used effectively to convey deeper meanings, such as the significance of family connections and the power of memories.

A.3.2 Evaluating Quality f_Q with GPT-4o

The evaluations consists in **score** + **brief explanation** for each criteria **A** to **O** (see 3.5).

Here, we present a generic example to illustrate the output of the LLM '*Quality*' evaluation.

-
- A. **9** The plot is coherent with a clear timeline and logical progression, hence a high score.
- B. **7** While it has some original elements, the time travel and coming-of-age themes are common in cinema, so it scores moderately.
- C. **8** Characters exhibit consistent traits throughout the plot, particularly Leo and his parents, justifying a high score.
- D. **8** The emotional stakes are significant, especially regarding family dynamics and personal growth, leading to a strong score.
- E. **6** The plot structure is moderately complex with time travel and character arcs, but not overly intricate, hence a middle score.
- F. **7** Characters show development, particularly Leo and his father, but some characters could be deeper, resulting in a solid score.
- G. **8** The world-building is rich, with a detailed setting and backstory that enhances the narrative, justifying a high score.
- H. **7** The pacing is generally good, though some scenes may feel slower, leading to a slightly lower score.
- I. **6** The themes of ambition, family, and overcoming the past are somewhat familiar, resulting in a moderate score.
- J. **9** The narrative flows well, with clear connections between events, justifying a high score.
- K. **5** While there are twists, some may be predictable, leading to a moderate score.
- L. **8** The potential for visually captivating scenes, especially with the time machine and storm, supports a high score.
- M. **7** The plot contains relevant social commentary about innovation and family dynamics, leading to a good score.
- N. **8** The themes explored are relevant to contemporary audiences, particularly regarding personal growth and societal change, justifying a high score.
- O. **7** The use of symbolism, particularly with the time machine and the clock tower, adds depth, leading to a solid score.

A.3.3 Evaluating Inspirations f_{insp} with GPT-3.5

Evaluates the similarity between the generated movie (Plot 1) and **Back to the Future** (Plot 2)

A. The setting and environment:

- Plot 1: Set in the small town of Valley Hills in the year 2025, with time travel technology and government regulations.
- Plot 2: Set in Hill Valley, California, with a time-traveling DeLorean and various time periods visited.

B. The thematic explored and issues covered:

- Plot 1: Focuses on personal growth, the consequences of altering the past, and the power of love and determination.
- Plot 2: Involves themes of family dynamics, altering the past to secure a better future, and the impact of individual actions

on the timeline.

C. The personality traits of the characters:

- Plot 1: Features characters like Alex, Dr. Elena, Agent Black, Sarah, and Mr. and Mrs. Reed with specific traits and roles.
- Plot 2: Includes Marty McFly, George McFly, Lorraine Baines McFly, Doc Brown, Biff Tannen, Jennifer, Dave, and Linda with

their distinct personalities and relationships.

D. The most important events:

- Plot 1: Involves Alex's time-traveling journey, altering his parents' lives, facing consequences, and making sacrifices to

restore the timeline.

- Plot 2: Revolves around Marty's accidental time travel to 1955, ensuring his parents' meeting, and securing a better future

for his family.

Comparing the two plots:

- Setting and Environment: **3** (Both involve time travel but in different contexts and locations.)
- Thematic Exploration: **4** (While both involve altering the past and personal growth, the specific themes and messages differ.)
- Character Traits: **2** (The characters in both plots have unique traits, but their roles and development are distinct.)
- Important Events: **3** (Both plots have significant events related to time travel and altering the past, but the details and

outcomes vary.)

A.3.4 Evaluating Inspirations f_{insp} with GPT-4o

Evaluates the similarity between the generated movie and **Back to the Future**

A. The setting and environment: Both plots are set in Hill Valley, California, with the first plot incorporating elements of time travel to 1955 and a focus on toys, while the second plot primarily revolves around the contemporary and historical aspects of the town, particularly in the 1980s and 1950s.

B. The thematic explored and issues covered: The first plot explores themes of friendship, teamwork, and personal growth through the adventures of toys and a teenager, while the second plot delves into family dynamics, self-identity, and the consequences of actions across time.

C. The personality traits of the characters: In the first plot, Woody is brave and caring, Buzz is confident, and Marty is a dreamer who learns to face his fears. In the second plot, Marty is ambitious but initially underachieving, George is timid, and Lorraine is unhappy. There are some overlapping traits of ambition and personal growth, but the characters' contexts and developments differ.

D. The most important events: The first plot involves toys working together to help their owner and a teenager fix his family issues, while the second plot centers around Marty traveling back in time to ensure his parents fall in love and secure his own existence. Both involve significant challenges and resolutions, but the specifics of the events are quite different.

Setting and Environment: **9** (Both plots are set in Hill Valley, California, with time travel elements in both.)

Thematic Exploration: **5** (Both explore themes of personal growth and relationships, but the contexts differ significantly.)

Character Traits: **4** (Some similarities in ambition and personal growth, but the characters are distinct in their roles and development.)

Important Events: **3** (Both involve significant challenges and resolutions, but the events and contexts are quite different.)

A.3.5 Evaluating Instructions f_{inst} with GPT-3.5

Evaluates how well the generated movie follows the **user guidelines**

Instructions:

A. A young girl discovers her old toys have the ability to travel through time

B. Heartwarming adventure leading to an adventure where she reunites with her young parents in the past to uncover family secrets

Evaluation and justification:

A. ****8**** (The plot includes the element of toys with the ability to travel through time)

B. ****9**** (The plot involves a heartwarming adventure where the young girl reunites with her parents in the past to uncover family secrets. The emotional aspect and family reunion are well portrayed)

A.3.6 Evaluating Instructions f_{inst} with GPT-4o

Evaluates how well the generated movie follows the **user guidelines**

Instructions:

- A young girl discovers her old toys have the ability to travel through time

- The story leads to an adventure where she reunites with her young parents in the past

- The plot involves uncovering family secrets

Evaluation and justification:

1. A young girl discovers her old toys have the ability to travel through time: **2** (The protagonist is Woody, a toy, and the main character is Marty, a teenager, not a young girl.)

2. The story leads to an adventure where she reunites with her young parents in the past: **8** (Marty, the main character, does reunite with his young parents in the past, but the focus is not on a young girl.)

3. The plot involves uncovering family secrets: **5** (While Marty tries to keep his parents together, the plot does not explicitly focus on uncovering family secrets.)

Overall, the plot does not fully align with the instructions, particularly regarding the main character being a young girl. The adventure and time travel elements are present, but the focus on family secrets is not as strong as it could be.

